WEST

| Help | Logout | Interrupt |

| Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases |

Search Results -

Terms	Documents
L3 AND list	38

Database:

- US Patents Full-Text Database
- US Pre-Grant Publication Full-Text Database
- JPO Abstracts Database
- EPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Search: L4

[Refine Search]

[Recall Text] [Clear]

Search History

DATE: Sunday, February 23, 2003   Printable Copy   Create Case

Set Name Query	Hit Count	Set Name
side by side		result set
DB=USPT; PLUR=NO; OP=OR		
L4 L3 AND list	38	L4
L3 L2 AND selection	41	L3
L2 (Software ADJ distribution) AND ((717/$$$)!.CCLS.)	85	L2
L1 (5826270 5890163 6292830 6041572 6253193 5931909 5721824 5950010 5253331 5586322 5634016 5671412 5832511 5852714 5857967 6006035 6014760 6123737 6163805 6167537 6223345 6223345 6236989 6263491 6266811 6282711 5764981 5289574 5784563 5440739 5761673 6101325 6133626 6211575 6211575 6239797 6332217 6182056 5699244 5586304 5588143 5809287 5960189 6141681 6175839 6321334 5819093 6042614 6105073 5600833).dwku.	0	L1

END OF SEARCH HISTORY

WEST Search History

DATE: Sunday, February 23, 2003

Set Name Query	Hit Count	Set Name
side by side		result set

DB=USPT; PLUR=NO; OP=OR

Set Name	Query	Hit Count	Set Name
L4	L3 AND list	38	L4
L3	L2 AND selection	41	L3
L2	(Software ADJ distribution) AND ((717/$$$)!.CCLS.)	85	L2
L1	(5826270 5890163 6292830 6041572 6253193 5931909 5721824 5950010 5253331 5586322 5634016 5671412 5832511 5852714 5857967 6006035 6014760 6123737 6163805 6167537 6223345 6223345 6236989 6263491 6266811 6282711 5764981 5289574 5784563 5440739 5761673 6101325 6133626 6211575 6211575 6239797 6332217 6182056 5699244 5586304 5588143 5809287 5960189 6141681 6175839 6321334 5819093 6042614 6105073 5600833).dwku.	0	L1

END OF SEARCH HISTORY

FreeBSD Handbook

Preface

Intended Audience

The FreeBSD newcomer will find that the first section of this book guides the user through the FreeBSD installation process, and gently introduces the concepts and conventions that underpin Unix. Working through this section requires little more than the desire to explore, and the ability to take on board new concepts as they are introduced.

Once you have travelled this far, the second, far larger, section of the Handbook is a comprehensive reference to all manner of topics of interest to FreeBSD system administrators. Some of these chapters may recommend that you do some prior reading, and this is noted in the synopsis at the beginning of each chapter.

For a list of additional sources of information, please see <u>Appendix B</u>.

Changes from the First Edition

This second edition is the culmination of over two years of work by the dedicated members of the FreeBSD Documentation Project. The following are the major changes in this new edition:

- A complete Index has been added.

- All ASCII figures have been replaced by graphical diagrams.

- A standard synopsis has been added to each chapter to give a quick summary of what information the chapter contains, and what the reader is expected to know.

- The content has been logically reorganized into three parts: ``Getting Started", ``System Administration", and ``Appendices".

- <u>Chapter 2</u> (``Installing FreeBSD") was completely rewritten with many screenshots to make it much easier for new users to grasp the text.

- Chapter 3 (``Unix Basics'') has been expanded to contain additional information about processes, daemons, and signals.

- Chapter 4 (``Installing Applications'') has been expanded to contain additional information about binary package management.

- Chapter 5 (``The X Window System'') has been completely rewritten with an emphasis on using modern desktop technologies such as **KDE** and **GNOME** on XFree86 4.X.

- Chapter 7 (``The FreeBSD Booting Process'') has been expanded.

- Chapter 12 (``Storage'') has been written from what used to be two separate chapters on ``Disks'' and ``Backups''. We feel that the topics are easier to comprehend when presented as a single chapter. A section on RAID (both hardware and software) has also been added.

- Chapter 17 (``Serial Communications'') has been completely reorganized and updated for FreeBSD 4.X/5.X.

- Chapter 18 (``PPP and SLIP'') has been substantially updated.

- Many new sections have been added to Chapter 19 (``Advanced Networking'').

- Chapter 20 (``Electronic Mail'') has been expanded to include more information about configuring **sendmail**.

- Chapter 22 (``Linux Compatibility'') has been expanded to include information about installing **Oracle** and **SAP/R3**.

- The following new topics are covered in this second edition:

    - Configuration and Tuning (Chapter 6).

    - Multimedia (Chapter 16)

Organization of This Book

This book is split into three logically distinct sections. The first section, *Getting Started*, covers the installation and basic usage of FreeBSD. It is expected that the reader will follow these chapters in sequence, possibly skipping chapters covering familiar topics. The second section, *System Administration*, covers a broad collection of subjects that are of interest to more advanced FreeBSD users. Each section begins with a succinct synopsis that describes what the chapter covers and what the reader is expected to already know. This is meant to allow the casual reader to skip around to find chapters of interest. The third section contains appendices of reference information.

Chapter 1, *Introduction*

Introduces FreeBSD to a new user. It describes the history of the FreeBSD Project, its goals and development model.

Chapter 2, *Installation*

Walks a user through the entire installation process. Some advanced installation topics, such as installing through a serial console, are also covered.

Chapter 3, *Unix Basics*

Covers the basic commands and functionality of the FreeBSD operating system. If you are familiar with Linux or another flavor of Unix then you can probably skip this chapter.

Chapter 4, *Installing Applications*

Covers the installation of third-party software with both FreeBSD's innovative ``Ports Collection" and standard binary packages.

Chapter 5, *The X Window System*

Describes the X Window System in general and using **XFree86** on FreeBSD in particular. Also describes common desktop environments such as **KDE** and **GNOME**.

Chapter 6, *Configuration and Tuning*

Describes the parameters available for system administrators to tune a FreeBSD system for optimum performance. Also describes the various configuration files used in FreeBSD and where to find them.

Chapter 7, *Booting Process*

Describes the FreeBSD boot process and explains how to control this process with configuration options.

Chapter 8, *Users and Basic Account Management*

Describes the creation and manipulation of user accounts. Also discusses resource limitations that can be set on users and other account management tasks.

Chapter 9, *Configuring the FreeBSD Kernel*

Explains why you might need to configure a new kernel and provides detailed instructions for configuring, building, and installing a custom kernel.

Chapter 10, *Security*

Describes many different tools available to help keep your FreeBSD system secure, including Kerberos, IPsec, OpenSSH, and network firewalls.

Chapter 11, Printing

Describes managing printers on FreeBSD, including information about banner pages, printer accounting, and initial setup.

Chapter 12, Storage

Describes how to manage storage media and filesystems with FreeBSD. This includes physical disks, RAID arrays, optical and tape media, memory-backed disks, and network filesystems.

Chapter 13, Vinum

Describes how to use Vinum, a logical volume manager which provides device-independent logical disks, and software RAID-0, RAID-1 and RAID-5.

Chapter 14, Localization

Describes how to use FreeBSD in languages other than English. Covers both system and application level localization.

Chapter 15, Desktop Applications

Lists some common desktop applications, such as web browsers and productivity suites, and describes how to install them on FreeBSD.

Chapter 16, Multimedia

Shows how to setup sound and video playback support for your system. Also describes some sample audio and video applications.

Chapter 17, Serial Communications

Explains how to connect terminals and modems to your FreeBSD system for both dial in and dial out connections.

Chapter 18, PPP and SLIP

Describes how to use PPP, SLIP, or PPP over Ethernet to connect to remote systems with FreeBSD.

Chapter 19, Advanced Networking

Describes many networking topics, including sharing an Internet connection with other computers on your LAN, using network filesystems, sharing account information via NIS, setting up a name server, and much more.

Chapter 20, Electronic Mail

Explains the different components of an email server and dives into simple configuration topics for the most popular mail server software: **sendmail**.

Chapter 21, The Cutting Edge

Explains the differences between FreeBSD-STABLE, FreeBSD-CURRENT, and FreeBSD releases. Describes which users would benefit from tracking a development system and outlines that process.

Chapter 22, Linux Binary Compatibility

Describes the Linux compatibility features of FreeBSD. Also provides detailed installation instructions for many popular Linux applications such as **Oracle, SAP/R3**, and **Mathematica**.

Appendix A, Obtaining FreeBSD

Lists different sources for obtaining FreeBSD media on CDROM or DVD as well as different sites on the Internet that allow you to download and install FreeBSD.

Appendix B, Bibliography

This book touches on many different subjects that may leave you hungry for a more detailed explanation. The bibliography lists many excellent books that are referenced in the text.

Appendix C, Resources on the Internet

Describes the many forums available for FreeBSD users to post questions and engage in technical conversations about FreeBSD.

Appendix D, PGP Keys

Lists the PGP fingerprints of several FreeBSD Developers.

Conventions used in this book

To provide a consistent and easy to read text, several conventions are followed throughout the book.

Typographic Conventions

Italic

An _italic_ font is used for filenames, URLs, emphasized text, and the first usage of technical terms.

`Monospace`

A monospaced font is used for error messages, commands, environment variables, names of ports, hostnames, user names, group names, device names, variables, and code fragments.

Bold

> A **bold** font is used for applications, commands, and keys.

User Input

Keys are rendered in **bold** to stand out from other text. Key combinations that are meant to be typed simultaneously are rendered with `+' between the keys, such as:

Ctrl+Alt+Del

Keys that are meant to be typed in sequence will be separated with commas, for example:

Ctrl+X, Ctrl+S

Would mean that the user is expected to type the **Ctrl** and **X** keys simultaneously and then to type the **Ctrl** and **S** keys simultaneously.

Examples

Examples starting with E:\> indicate a MS-DOS command. Unless otherwise noted, these commands may be executed from a ``Command Prompt'' window in a modern Microsoft Windows environment.

```
E:\> tools\fdimage floppies\kern.flp A:
```

Examples starting with # indicate a command that must be invoked as the superuser in FreeBSD. You can login as root to type the command, or login as your normal account and use su(1) to gain superuser privileges.

```
# dd if=kern.flp of=/dev/fd0
```

Examples starting with % indicate a command that should be invoked from a normal user account. Unless otherwise noted, C-shell syntax is used for setting environment variables and other shell commands.

`% top`

Acknowledgments

The book you are holding represents the efforts of many hundreds of people around the world. Whether they sent in fixes for typos, or submitted complete chapters, all the contributions have been useful.

Several companies have supported the development of this document by paying authors to work on it full-time, paying for publication, etc. In particular, BSDi (subsequently acquired by Wind River Systems) paid members of the FreeBSD Documentation Project to work on improving this book full time leading up to the publication of the first printed edition in March 2000 (ISBN 1-57176-241-8). Wind River Systems then paid several additional authors to make a number of improvements to the print-output infrastructure and to add additional chapters to the text. This work culminated in the publication of the second printed edition in November 2001 (ISBN 1-57176-303-1).

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

For questions about FreeBSD, read the documentation before contacting <questions@FreeBSD.org>.
For questions about this documentation, e-mail <doc@FreeBSD.org>.

FreeBSD Handbook

I. Getting Started

This part of the FreeBSD Handbook is for users and administrators who are new to FreeBSD. These chapters:

- Introduce you to FreeBSD.

- Guide you through the installation process.

- Teach you some Unix basics.

- Show you how to install the wealth of third party applications available for FreeBSD.

- Introduce you to X, the Unix windowing system, and detail how to configure a desktop environment that makes you more productive.

We have tried to keep the number of forward references in the text to a minimum so that you can read this section of the Handbook from front to back with the minimum of page flipping required.

Table of Contents

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

For questions about FreeBSD, read the documentation before contacting <questions@FreeBSD.org>.
For questions about this documentation, e-mail <doc@FreeBSD.org>.

FreeBSD Handbook

Chapter 1 Introduction

Table of Contents

Restructured, reorganized, and parts rewritten by Jim Mock.

1.1 Synopsis

Thank you for your interest in FreeBSD! The following chapter covers various aspects of the FreeBSD Project, such as its history, goals, development model, and so on.

After reading this chapter, you will know:

- How FreeBSD relates to other computer operating systems.

- The history of the FreeBSD Project.

- The goals of the FreeBSD Project.

- The basics of the FreeBSD open-source development model.

- And of course: where the name ``FreeBSD'' comes from.

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

For questions about FreeBSD, read the documentation before contacting <questions@FreeBSD.org>.

For questions about this documentation, e-mail <doc@FreeBSD.org>.

FreeBSD Handbook

Chapter 1 Introduction

Table of Contents

Restructured, reorganized, and parts rewritten by Jim Mock.

1.1 Synopsis

Thank you for your interest in FreeBSD! The following chapter covers various aspects of the FreeBSD Project, such as its history, goals, development model, and so on.

After reading this chapter, you will know:

- How FreeBSD relates to other computer operating systems.

- The history of the FreeBSD Project.

- The goals of the FreeBSD Project.

- The basics of the FreeBSD open-source development model.

- And of course: where the name ``FreeBSD" comes from.

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

For questions about FreeBSD, read the documentation before contacting <questions@FreeBSD.org>.

For questions about this documentation, e-mail <doc@FreeBSD.org>.

FreeBSD Handbook

Chapter 1 Introduction

1.2 Welcome to FreeBSD!

FreeBSD is a 4.4BSD-Lite based operating system for the Intel architecture (x86) and DEC Alpha based systems. Ports to other architectures are also underway. For a brief overview of FreeBSD, see the next section. You can also read about the history of FreeBSD, or the current release. If you are interested in contributing something to the Project (code, hardware, unmarked bills), see the Contributing to FreeBSD article.

1.2.1 What Can FreeBSD Do?

FreeBSD has many noteworthy features. Some of these are:

- *Preemptive multitasking* with dynamic priority adjustment to ensure smooth and fair sharing of the computer between applications and users, even under the heaviest of loads.

- *Multi-user facilities* which allow many people to use a FreeBSD system simultaneously for a variety of things. This means, for example, that system peripherals such as printers and tape drives are properly shared between all users on the system or the network and that individual resource limits can be placed on users or groups of users, protecting critical system resources from over-use.

- Strong *TCP/IP networking* with support for industry standards such as SLIP, PPP, NFS, DHCP, and NIS. This means that your FreeBSD machine can interoperate easily with other systems as well as act as an enterprise server, providing vital functions such as NFS (remote file access) and email services or putting your organization on the Internet with WWW, FTP, routing and firewall (security) services.

- *Memory protection* ensures that applications (or users) cannot interfere with each other. One application crashing will not affect others in any way.

- FreeBSD is a *32-bit* operating system (*64-bit* on the Alpha) and was designed as such from the ground up.

- The industry standard *X Window System* (X11R6) provides a graphical user interface (GUI) for the cost of a common VGA card and monitor and comes with full sources.

- *Binary compatibility* with many programs built for Linux, SCO, SVR4, BSDI and NetBSD.

- Thousands of *ready-to-run* applications are available from the FreeBSD *ports* and *packages* collection. Why search the net when you can find it all right here?

- Thousands of additional and *easy-to-port* applications are available on the Internet. FreeBSD is source code compatible with most popular commercial Unix systems and thus most applications require few, if any, changes to compile.

- Demand paged *virtual memory* and ``merged VM/buffer cache'' design efficiently satisfies applications with large appetites for memory while still maintaining interactive response to other users.

- *SMP* support for machines with multiple CPUs.

- A full complement of *C*, *C++*, *Fortran*, and *Perl* development tools. Many additional languages for advanced research and development are also available in the ports and packages collection.

- *Source code* for the entire system means you have the greatest degree of control over your environment. Why be locked into a proprietary solution at the mercy of your vendor when you can have a truly open system?

- Extensive *online documentation*.

- *And many more!*

FreeBSD is based on the 4.4BSD-Lite release from Computer Systems Research Group (CSRG) at the University of California at Berkeley, and carries on the distinguished tradition of BSD systems development. In addition to the fine work provided by CSRG, the FreeBSD Project has put in many thousands of hours in fine tuning the system for maximum performance and reliability in real-life load situations. As many of the commercial giants struggle to field PC operating systems with such features, performance and reliability, FreeBSD can offer them *now*!

The applications to which FreeBSD can be put are truly limited only by your own imagination. From software development to factory automation, inventory control to azimuth correction of remote satellite antennae; if it can be done with a commercial Unix product then it is more than likely that you can do it with FreeBSD too! FreeBSD also benefits significantly from literally thousands of high quality applications developed by research centers and universities around the world, often available at little to no cost. Commercial applications are also available and appearing in greater numbers every day.

Because the source code for FreeBSD itself is generally available, the system can also be customized to an almost unheard of degree for special applications or projects, and in ways not generally possible with operating systems from most major commercial vendors. Here is just a sampling of some of the applications in which people are currently using FreeBSD:

- *Internet Services:* The robust TCP/IP networking built into FreeBSD makes it an ideal platform for a variety of Internet services such as:

  - FTP servers

  - World Wide Web servers (standard or secure [SSL])

  - Firewalls and NAT ("IP masquerading") gateways

  - Electronic Mail servers

  - USENET News or Bulletin Board Systems

  - And more...

  With FreeBSD, you can easily start out small with an inexpensive 386 class PC and upgrade all the way up to a quad-processor Xeon with RAID storage as your enterprise grows.

- *Education:* Are you a student of computer science or a related engineering field? There is no better way of learning about operating systems, computer architecture and networking than the hands on, under the hood experience that FreeBSD can provide. A number of freely available CAD, mathematical and graphic design packages also make it highly useful to those whose primary interest in a computer is to get *other* work done!

- *Research:* With source code for the entire system available, FreeBSD is an excellent platform for research in operating systems as well as other branches of computer science. FreeBSD's freely available nature also makes it possible for remote groups to collaborate on ideas or shared development without having to worry about special licensing agreements or limitations on what may be discussed in open forums.

- *Networking:* Need a new router? A name server (DNS)? A firewall to keep people out of your internal network? FreeBSD can easily turn that unused 386 or 486 PC sitting in the corner into an advanced router with sophisticated packet-filtering capabilities.

- *X Window workstation:* FreeBSD is a fine choice for an inexpensive X terminal solution, either using the freely available XFree86 server or one of the excellent commercial servers provided by X Inside. Unlike an X terminal, FreeBSD allows many applications to be run locally if desired, thus relieving the burden on a central server. FreeBSD can even boot "diskless", making individual workstations even cheaper and easier to administer.

- *Software Development:* The basic FreeBSD system comes with a full complement of development tools including the renowned GNU C/C++ compiler and debugger.

FreeBSD is available in both source and binary form on CDROM and via anonymous FTP. Please see Appendix A for more information about obtaining FreeBSD.

1.2.2 Who uses FreeBSD?

FreeBSD is used to power some of the biggest sites on the Internet, including:

- Yahoo!

- Apache

- Blue Mountain Arts

- Pair Networks

- Sony Japan

- Netcraft

- Weathernews

- Supervalu

- TELEHOUSE America

- Sophos Anti-Virus

- JMA Wired

and many more.

Prev	Home	Next
Introduction	Up	About the FreeBSD Project

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

For questions about FreeBSD, read the documentation before contacting <questions@FreeBSD.org>.
For questions about this documentation, e-mail <doc@FreeBSD.org>.

What you'll find in this directory:

bootinst.exe   Install a boot manager.   Uses the boot.bin image.
boot.bin

ckdist.exe   Check the integrity of a distribution you've xferred.

fdimage.exe   Write disk images to floppy disk (supercedes rawrite).

fips.exe   Break FAT16 partitions into smaller pieces.

gunzip.exe   Unzip stuff.

ide_conf.exe   Display IDE configuration parameters, like geometry.

osbs135.exe   Version 1.35 and 2.0BETA of the OS-BS boot manager
osbsbeta.exe   (an alternative boot manager to BOOTEASY).

presizer.exe   Break FAT16/FAT32 partitions into smaller pieces.

rawrite.exe   Older utility for writing disk images (use fdimage).

restorrb.exe   Restore a boot block spammed by bootinst or OSBS.

```
/////////////////////////////////////////////////////////////////////////
/                             VI REFERENCE                               /
/////////////////////////////////////////////////////////////////////////
```

Warning: some vi versions don't support the more esoteric features
described in this document. You can edit/redistribute this document
freely, as long as you don't make false claims on original authorship.

Author: Maarten Litmaath <maart@nat.vu.nl>
Version: 8

```
//////////////////
/ contributions /
//////////////////
```

Rich Salz <rsalz@bbn.com>
Eamonn McManus <emcmanus@cs.tcd.ie>
Diomidis Spinellis <diomidis%ecrcvax.uucp@pyramid.pyramid.com>
Blair P. Houghton <bph@buengc.bu.edu>
Rusty Haddock <{uunet,att,rutgers}!mimsy.umd.edu!fe2o3!rusty>
Panos Tsirigotis <panos@boulder.colorado.edu>
David J. MacKenzie <djm@wam.umd.edu>
Kevin Carothers <kevin@ttidca.tti.com>
Dan Mercer <mercer@ncrcce.StPaul.NCR.COM>
Ze'ev Shtadler <steed@il4cad.intel.com>
Paul Quare <pq@r2.cs.man.ac.uk>
Dave Beyerl <att!ihlpl!db21>
Lee Sailer <UH2@psuvm.psu.edu>
David Gast <gast@cs.ucla.edu>

```
//////////////
/ legenda /
//////////////
```

```
default values    : 1
<*>               : `*' must not be taken literally
[*]               : `*' is optional
^X                : <ctrl>X
<sp>              : space
<cr>              : carriage return
<lf>              : linefeed
<ht>              : horizontal tab
<esc>             : escape
<erase>           : your erase character
<kill>            : your kill character
<intr>            : your interrupt character
<a-z>             : an element in the range
N                 : number (`*' = allowed, `-' = not appropriate)
CHAR              : char unequal to <ht>|<sp>
WORD              : word followed by <ht>|<sp>|<lf>
```

```
///////////////////
/ move commands /
///////////////////
```

```
N | Command               | Meaning
--+----------------------+-------------------------------------------------------
* | h | ^H | <erase>      | <*> chars to the left.
* | j | <lf> | ^N         | <*> lines downward.
* | l | <sp>              | <*> chars to the right.
* | k | ^P                | <*> lines upward.
* | $                     | To the end of line <*> from the cursor.
- | ^                     | To the first CHAR of the line.
* | -                     | To the first CHAR <*> - 1 lines lower.
* | _                     | To the first CHAR <*> lines higher.
* | + | <cr>              | To the first CHAR <*> lines lower.
- | 0                     | To the first char of the line.
* | |                     | To column <*> (<ht>: only to the endpoint).
* | f<char>               | <*> <char>s to the right (find).
* | t<char>               | Till before <*> <char>s to the right.
* | F<char>               | <*> <char>s to the left.
* | T<char>               | Till after <*> <char>s to the left.
* | ;                     | Repeat latest `f'|`t'|`F'|`T' <*> times.
* | ,                     | Idem in opposite direction.
* | w                     | <*> words forward.
* | W                     | <*> WORDS forward.
* | b                     | <*> words backward.
* | B                     | <*> WORDS backward.
* | e                     | To the end of word <*> forward.
* | E                     | To the end of WORD <*> forward.
* | G                     | Go to line <*> (default EOF).
* | H                     | To line <*> from top of the screen (home).
* | L                     | To line <*> from bottom of the screen (last).
- | M                     | To the middle line of the screen.
* | )                     | <*> sentences forward.
* | (                     | <*> sentences backward.
* | }                     | <*> paragraphs forward.
* | {                     | <*> paragraphs backward.
- | ]]                    | To the next section (default EOF).
- | [[                    | To the previous section (default begin of file).
- | `<a-z>                | To the mark.
- | '<a-z>                | To the first CHAR of the line with the mark.
- | ``                    | To the cursor position before the latest absolute
  |                       |   jump (of which are examples `/' and `G').
- | ''                    | To the first CHAR of the line on which the cursor
  |                       |   was placed before the latest absolute jump.
- | /<string>             | To the next occurrence of <string>.
- | ?<string>             | To the previous occurrence of <string>.
- | n                     | Repeat latest `/'|`?' (next).
- | N                     | Idem in opposite direction.
- | %                     | Find the next bracket and go to its match
  |                       |   (also with `{'|`}' and `['|`]').
```

```
/////////////////////////
/ searching (see above) /
/////////////////////////
```

```
:ta <name>              | Search in the tags file[s] where <name> is
                        |   defined (file, line), and go to it.
^]                      | Use the name under the cursor in a `:ta' command.
^T                      | Pop the previous tag off the tagstack and return
                        |   to its position.
:[x,y]g/<string>/<cmd>  | Search globally [from line x to y] for <string>
                        |   and execute the `ex' <cmd> on each occurrence.
:[x,y]v/<string>/<cmd>  | Execute <cmd> on the lines that don't match.
```

```
/////////////////////
/ undoing changes /
/////////////////////

u                             | Undo the latest change.
U                             | Undo all changes on a line, while not having
                              |   moved off it (unfortunately).
:q!                           | Quit vi without writing.
:e!                           | Re-edit a messed-up file.


/////////////////////////////////////
/ appending text (end with <esc>) /
/////////////////////////////////////

* | a                         | <*> times after the cursor.
* | A                         | <*> times at the end of line.
* | i                         | <*> times before the cursor (insert).
* | I                         | <*> times before the first CHAR of the line
* | o                         | On a new line below the current (open).
                              |   The count is only useful on a slow terminal.
* | O                         | On a new line above the current.
                              |   The count is only useful on a slow terminal.
* | ><move>                   | Shift the lines described by <*><move> one
                              |   shiftwidth to the right.
* | >>                        | Shift <*> lines one shiftwidth to the right.
* | ["<a-zA-Z1-9>]p           | Put the contents of the (default undo) buffer
                              |   <*> times after the cursor.
                              |   A buffer containing lines is put only once,
                              |   below the current line.
* | ["<a-zA-Z1-9>]P           | Put the contents of the (default undo) buffer
                              |   <*> times before the cursor.
                              |   A buffer containing lines is put only once,
                              |   above the current line.
* | .                         | Repeat previous command <*> times.  If the last
                              |   command before a `.' command references a
                              |   numbered buffer, the buffer number is
                              |   incremented first (and the count is ignored):
                              |
                              |   "1pu.u.u.u      - `walk through' buffers 1
                              |                      through 5
                              |   "1P....         - restore them


/////////////////////
/ deleting text /
/////////////////////

Everything deleted can be stored into a buffer. This is achieved by
putting a `"' and a letter <a-z> before the delete command. The
deleted text will be in the buffer with the used letter. If <A-Z>
is used as buffer name, the conjugate buffer <a-z> will be augmented
instead of overwritten with the text. The undo buffer always
contains the latest change. Buffers <1-9> contain the latest 9
LINE deletions (`"1' is most recent).

* | x                         | Delete <*> chars under and after the cursor.
* | X                         | <*> chars before the cursor.
* | d<move>                   | From begin to endpoint of <*><move>.
* | dd                        | <*> lines.
- | D                         | The rest of the line.
```

```
* | <<move>         | Shift the lines described by <*><move> one
                     |    shiftwidth to the left.
* | <<              | Shift <*> lines one shiftwidth to the left.
* | .               | Repeat latest command <*> times.
```

```
/////////////////////////////////
/ changing text (end with <esc>) /
/////////////////////////////////
```

```
* | r<char>         | Replace <*> chars by <char> - no <esc>.
* | R               | Overwrite the rest of the line,
                    |    appending change <*> - 1 times.
* | s               | Substitute <*> chars.
* | S               | <*> lines.
* | c<move>         | Change from begin to endpoint of <*><move>.
* | cc              | <*> lines.
* | C               | The rest of the line and <*> - 1 next lines.
* | =<move>         | If the option `lisp' is set, this command
                    |    will realign the lines described by <*><move>
                    |    as though they had been typed with the option
                    |    `ai' set too.
- | ~               | Switch lower and upper cases
                    |    (should be an operator, like `c').
* | J               | Join <*> lines (default 2).
* | .               | Repeat latest command <*> times (`J' only once).
- | &               | Repeat latest `ex' substitute command, e.g.
                    |    `:s/wrong/good'.
- | :[x,y]s/<p>/<r>/<f>| Substitute (on lines x through y) the pattern <p>
                    |    (default the last pattern) with <r>.  Useful
                    |    flags <f> are `g' for `global' (i.e. change
                    |    every non-overlapping occurrence of <p>) and
                    |    `c' for `confirm' (type `y' to confirm a
                    |    particular substitution, else <cr>).  Instead
                    |    of `/' any punctuation CHAR unequal to <lf>
                    |    can be used as delimiter.
```

```
/////////////////////////////////
/ substitute replacement patterns /
/////////////////////////////////
```

The basic meta-characters for the replacement pattern are `&' and `~';
these are given as `\&' and `\~' when nomagic is set. Each instance
of `&' is replaced by the characters which the regular expression
matched. The meta-character `~' stands, in the replacement
pattern, for the defining text of the previous replacement
pattern. Other meta-sequences possible in the replacement pattern
are always introduced by the escaping character `\'. The sequence
`\n' (with `n' in [1-9]) is replaced by the text matched by the
n-th regular subexpression enclosed between `\(' and `\)'. The
sequences `\u' and `\l' cause the immediately following character
in the replacement to be converted to upper- or lower-case
respectively if this character is a letter. The sequences `\U' and
`\L' turn such conversion on, either until `\E' or `\e' is
encountered, or until the end of the replacement pattern.

```
/////////////////////////////////
/ remembering text (yanking) /
/////////////////////////////////
```

With yank commands you can put `"<a-zA-Z>' before the command, just as

with delete commands. Otherwise you only copy to the undo buffer.
The use of buffers <a-z> is THE way of copying text to another file;
see the `:e <file>' command.

```
* | y<move>            | Yank from begin to endpoint of <*><move>.
* | yy                 | <*> lines.
* | Y                  | Idem (should be equivalent to `y$' though).
- | m<a-z>             | Mark the cursor position with a letter.
```

```
/////////////////////////////////////
/ commands while in append|change mode /
/////////////////////////////////////
```

```
^@                     | If typed as the first character of the
                       |   insertion, it is replaced with the previous
                       |   text inserted (max. 128 chars), after which
                       |   the insertion is terminated.
^V                     | Deprive the next char of its special meaning
                       |   (e.g. <esc>).
^D                     | One shiftwidth to the left, but only if
                       |   nothing else has been typed on the line.
0^D                    | Remove all indentation on the current line
                       |   (there must be no other chars on the line).
^^D                    | Idem, but it is restored on the next line.
^T                     | One shiftwidth to the right, but only if
                       |   nothing else has been typed on the line.
^H | <erase>           | One char back.
^W                     | One word back.
<kill>                 | Back to the begin of the change on the
                       |   current line.
<intr>                 | Like <esc> (but you get a beep as well).
```

```
/////////////////////////////////////////////////
/ writing, editing other files, and quitting vi /
/////////////////////////////////////////////////
```

In `:' `ex' commands - if not the first CHAR on the line - `%' denotes
the current file, `#' is a synonym for the alternate file (which
normally is the previous file). As first CHAR on the line `%' is a
shorthand for `1,$'. Marks can be used for line numbers too: '<a-z>.
In the `:w'|`:f'|`:cd'|`:e'|`:n' commands shell meta-characters can be
used.

```
:q                     | Quit vi, unless the buffer has been changed.
:q!                    | Quit vi without writing.
^Z                     | Suspend vi.
:w                     | Write the file.
:w <name>              | Write to the file <name>.
:w >> <name>           | Append the buffer to the file <name>.
:w! <name>             | Overwrite the file <name>.
:x,y w <name>          | Write lines x through y to the file <name>.
:wq                    | Write the file and quit vi; some versions quit
                       |   even if the write was unsuccessful!
                       |   Use `ZZ' instead.
ZZ                     | Write if the buffer has been changed, and
                       |   quit vi.  If you have invoked vi with the `-r'
                       |   option, you'd better write the file
                       |   explicitly (`w' or `w!'), or quit the
                       |   editor explicitly (`q!') if you don't want
                       |   to overwrite the file - some versions of vi
```

```
                          | don't handle the `recover' option very well.
:x [<file>]               | Idem [but write to <file>].
:x! [<file>]              | `:w![<file>]' and `:q'.
:pre                      | Preserve the file - the buffer is saved as if
                          |   the system had just crashed; for emergencies,
                          |   when a `:w' command has failed and you don't
                          |   know how to save your work (see `vi -r').
:f <name>                 | Set the current filename to <name>.
:cd [<dir>]               | Set the working directory to <dir>
                          |   (default home directory).
:cd! [<dir>]              | Idem, but don't save changes.
:e [+<cmd>] <file>        | Edit another file without quitting vi - the
                          |   buffers are not changed (except the undo
                          |   buffer), so text can be copied from one file to
                          |   another this way.  [Execute the `ex' command
                          |   <cmd> (default `$') when the new file has been
                          |   read into the buffer.]  <cmd> must contain no
                          |   <sp> or <ht>.  See `vi startup'.
:e! [+<cmd>] <file>       | Idem, without writing the current buffer.
^^                        | Edit the alternate (normally the previous) file.
:rew                      | Rewind the argument list, edit the first file.
:rew!                     | Idem, without writing the current buffer.
:n [+<cmd>] [<files>]     | Edit next file or specify a new argument list.
:n! [+<cmd>] [<files>]    | Idem, without writing the current buffer.
:args                     | Give the argument list, with the current file
                          |   between `[' and `]'.
```

```
////////////////////
/ display commands /
////////////////////
```

```
^G                        | Give file name, status, current line number
                          |   and relative position.
^L                        | Refresh the screen (sometimes `^P' or `^R').
^R                        | Sometimes vi replaces a deleted line by a `@',
                          |   to be deleted by `^R' (see option `redraw').
[*]^E                     | Expose <*> more lines at bottom, cursor
                          |   stays put (if possible).
[*]^Y                     | Expose <*> more lines at top, cursor
                          |   stays put (if possible).
[*]^D                     | Scroll <*> lines downward
                          |   (default the number of the previous scroll;
                          |   initialization: half a page).
[*]^U                     | Scroll <*> lines upward
                          |   (default the number of the previous scroll;
                          |   initialization: half a page).
[*]^F                     | <*> pages forward.
[*]^B                     | <*> pages backward (in older versions `^B' only
                          |   works without count).
```

If in the next commands the field <wi> is present, the windowsize
will change to <wi>. The window will always be displayed at the
bottom of the screen.

```
[*]z[wi]<cr>              | Put line <*> at the top of the window
                          |   (default the current line).
[*]z[wi]+                 | Put line <*> at the top of the window
                          |   (default the first line of the next page).
[*]z[wi]-                 | Put line <*> at the bottom of the window
                          |   (default the current line).
```

```
[*]z[wi]^           | Put line <*> at the bottom of the window
                    |    (default the last line of the previous page).
[*]z[wi].           | Put line <*> in the centre of the window
                    |    (default the current line).
```

```
//////////////////////////
/ mapping and abbreviation /
//////////////////////////
```

When mapping take a look at the options `to' and `remap' (below).

```
:map <string> <seq>  | <string> is interpreted as <seq>, e.g.
                     |   `:map ^C :!cc %^V<cr>' to invoke `cc' (the C
                     |   compiler) from within the editor
                     |   (vi replaces `%' with the current file name).
:map                 | Show all mappings.
:unmap <string>      | Deprive <string> of its mapping.  When vi
                     |   complains about non-mapped macros (whereas no
                     |   typos have been made), first do something like
                     |   `:map <string> Z', followed by
                     |   `:unmap <string>' (`Z' must not be a macro
                     |   itself), or switch to `ex' mode first with `Q'.
:map! <string> <seq> | Mapping in append mode, e.g.
                     |   `:map! \be begin^V<cr>end;^V<esc>O<ht>'.
                     |   When in append mode <string> is preceded by
                     |   `^V', no mapping is done.
:map!                | Show all append mode mappings.
:unmap! <string>     | Deprive <string> of its mapping (see `:unmap').
:ab <string> <seq>   | Whenever in append mode <string> is preceded and
                     |   followed by a breakpoint (e.g. <sp> or `,'), it
                     |   is interpreted as <seq>, e.g.
                     |   `:ab ^P procedure'.  A `^V' immediately
                     |   following <string> inhibits expansion.
:ab                  | Show all abbreviations.
:unab <string>       | Do not consider <string> an abbreviation
                     |   anymore (see `:unmap').
@<a-z>               | Consider the contents of the named register a
                     |   command, e.g.:
                     |       o0^D:s/wrong/good/<esc>"zdd
                     |   Explanation:
                     |       o                - open a new line
                     |       0^D              - remove indentation
                     |       :s/wrong/good/ - this input text is an
                     |                          `ex' substitute command
                     |       <esc>            - finish the input
                     |       "zdd             - delete the line just
                     |                          created into register `z'
                     |   Now you can type `@z' to replace `wrong'
                     |   with `good' on the current line.
@@                   | Repeat last register command.
```

```
//////////////////////////
/ switch and shell commands /
//////////////////////////
```

```
Q | ^\ | <intr><intr>    | Switch from vi to `ex'.
:                        | An `ex' command can be given.
:vi                      | Switch from `ex' to vi.
:sh                      | Execute a subshell, back to vi by `^D'.
:[x,y]!<cmd>             | Execute a shell <cmd> [on lines x through y;
                         |    these lines will serve as input for <cmd> and
                         |    will be replaced by its standard output].
:[x,y]!! [<args>]        | Repeat last shell command [and append <args>].
:[x,y]!<cmd> ! [<args>]  | Use the previous command (the second `!') in a
                         |    new command.
[*]!<move><cmd>          | The shell executes <cmd>, with as standard
                         |    input the lines described by <*><move>,
                         |    next the standard output replaces those lines
                         |    (think of `cb', `sort', `nroff', etc.).
[*]!<move>!<args>        | Append <args> to the last <cmd> and execute it,
                         |    using the lines described by the current
                         |    <*><move>.
[*]!!<cmd>               | Give <*> lines as standard input to the
                         |    shell <cmd>, next let the standard output
                         |    replace those lines.
[*]!!! [<args>]          | Use the previous <cmd> [and append <args> to it].
:x,y w !<cmd>            | Let lines x to y be standard input for <cmd>
                         |    (notice the <sp> between the `w' and the `!').
:r!<cmd>                 | Put the output of <cmd> onto a new line.
:r <name>                | Read the file <name> into the buffer.
```

```
/////////////
/ vi startup /
/////////////
```

```
vi [<files>]             | Edit the files, start with the first page of
                         |    the first file.
```

The editor can be initialized by the shell variable `EXINIT', which looks like:

```
        EXINIT='<cmd>|<cmd>|...'
        <cmd>: set options
                map ...
                ab ...
        export EXINIT (in the Bourne shell)
```

However, the list of initializations can also be put into a file. If this file is located in your home directory, and is named `.exrc' AND the variable `EXINIT' is NOT set, the list will be executed automatically at startup time. However, vi will always execute the contents of a `.exrc' in the current directory, if you own the file. Else you have to give the execute (`source') command yourself:

```
        :so file
```

In a `.exrc' file a comment is introduced with a double quote character: the rest of the line is ignored.  Exception: if the last command on the line is a `map[!]' or `ab' command or a shell escape, a trailing comment is not recognized, but considered part of the command.

On-line initializations can be given with `vi +<cmd> file', e.g.:

```
vi +x file               | The cursor will immediately jump to line x
                         |    (default last line).
```

```
vi +/<string> file       | Jump to the first occurrence of <string>.

You can start at a particular tag with:

vi -t <tag>              | Start in the right file in the right place.
```

Sometimes (e.g. if the system crashed while you were editing) it is
possible to recover files lost in the editor by `vi -r file'. A plain
`vi -r' command shows the files you can recover.
If you just want to view a file by using vi, and you want to avoid any
change, instead of vi you can use the `view' or `vi -R' command:
the option `readonly' will be set automatically (with `:w!' you can
override this option).

```
/////////////////////////////
/ the most important options /
/////////////////////////////

ai                       | autoindent - In append mode after a <cr> the
                         |   cursor will move directly below the first
                         |   CHAR on the previous line.  However, if the
                         |   option `lisp' is set, the cursor will align
                         |   at the first argument to the last open list.
aw                       | autowrite - Write at every shell escape
                         |   (useful when compiling from within vi).
dir=<string>             | directory - The directory for vi to make
                         |   temporary files (default `/tmp').
eb                       | errorbells - Beeps when you goof
                         |   (not on every terminal).
ic                       | ignorecase - No distinction between upper and
                         |   lower cases when searching.
lisp                     | Redefine the following commands:
                         |   `(', `)'   - move backward (forward) over
                         |                 S-expressions
                         |   `{', `}'   - idem, but don't stop at atoms
                         |   `[[', `]]' - go to previous (next) line
                         |                 beginning with a `('
                         |   See option `ai'.
list                     | <lf> is shown as `$', <ht> as `^I'.
magic                    | If this option is set (default), the chars `.',
                         |   `[' and `*' have special meanings within search
                         |   and `ex' substitute commands.  To deprive such
                         |   a char of its special function it must be
                         |   preceded by a `\'.  If the option is turned off
                         |   it's just the other way around.  Meta-chars:
                         |   ^<string>     - <string> must begin the line
                         |   <string>$     - <string> must end the line
                         |   .             - matches any char
                         |   [a-z]         - matches any char in the range
                         |   [^a-z]        - any char not in the range
                         |   [<string>]    - matches any char in <string>
                         |   [^<string>]   - any char not in <string>
                         |   <char>*       - 0 or more <char>s
                         |   \<<string>    - <string> must begin a word
                         |   <string>\>    - <string> must end a word
modeline                 | When you read an existing file into the buffer,
                         |   and this option is set, the first and last 5
                         |   lines are checked for editing commands in the
                         |   following form:
                         |
```

```
                              |       <sp>vi:set options|map ...|ab ...|!...:
                              |
                              |  Instead of <sp> a <ht> can be used, instead of
                              |  `vi' there can be `ex'.  Warning: this option
                              |  could have nasty results if you edit a file
                              |  containing `strange' modelines.
nu                            | number - Numbers before the lines.




para=<string>                | paragraphs - Every pair of chars in <string> is
                             |    considered a paragraph delimiter nroff macro
                             |    (for `{' and `}').  A <sp> preceded by a `\'
                             |    indicates the previous char is a single letter
                             |    macro.  `:set para=P\ bp' introduces `.P' and
                             |    `.bp' as paragraph delimiters.  Empty lines and
                             |    section boundaries are paragraph boundaries
                             |    too.
redraw                       | The screen remains up to date.
remap                        | If on (default), macros are repeatedly
                             |    expanded until they are unchanged.
                             |    Example: if `o' is mapped to `A', and `A'
                             |    is mapped to `I', then `o' will map to `I'
                             |    if `remap' is set, else it will map to `A'.
report=<*>                   | Vi reports whenever e.g. a delete
                             |    or yank command affects <*> or more lines.
ro                           | readonly - The file is not to be changed.
                             |    However, `:w!' will override this option.
sect=<string>                | sections - Gives the section delimiters (for `[['
                             |    and `]]'); see option `para'. A `{' beginning a
                             |    line also starts a section (as in C functions).
sh=<string>                  | shell - The program to be used for shell escapes
                             |    (default `$SHELL' (default `/bin/sh')).
sw=<*>                       | shiftwidth - Gives the shiftwidth (default 8
                             |    positions).
sm                           | showmatch - Whenever you append a `)', vi shows
                             |    its match if it's on the same page; also with
                             |    `{' and `}'.  If there's no match at all, vi
                             |    will beep.
taglength=<*>                | The number of significant characters in tags
                             |    (0 = unlimited).
tags=<string>                | The space-separated list of tags files.
terse                        | Short error messages.
to                           | timeout - If this option is set, append mode
                             |    mappings will be interpreted only if they're
                             |    typed fast enough.
ts=<*>                       | tabstop - The length of a <ht>; warning: this is
                             |    only IN the editor, outside of it <ht>s have
                             |    their normal length (default 8 positions).
wa                           | writeany - No checks when writing (dangerous).
warn                         | Warn you when you try to quit without writing.
wi=<*>                       | window - The default number of lines vi shows.
wm=<*>                       | wrapmargin - In append mode vi automatically
                             |    puts a <lf> whenever there is a <sp> or <ht>
                             |    within <wm> columns from the right margin
                             |    (0 = don't put a <lf> in the file, yet put it
                             |    on the screen).
ws                           | wrapscan - When searching, the end is
                             |    considered `stuck' to the begin of the file.

:set <option>                | Turn <option> on.
:set no<option>              | Turn <option> off.
```

```
:set <option>=<value>    | Set <option> to <value>.
:set                     | Show all non-default options and their values.
:set <option>?           | Show <option>'s value.
:set all                 | Show all options and their values.
```
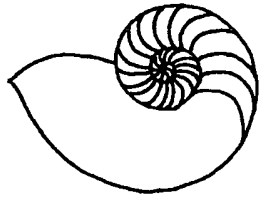
MUUG Lines

Manitoba UNIX®User Group

Newsletter of the Manitoba UNIX⁽⁾ User Group

MUUG Goes Online!
By Roland Schneider

For any of our members who don't know about the *MUUG Online* project yet, a little background is in order. The University of Manitoba Computer Centre has given MUUG access to a Sun386i workstation which they no longer require. The machine is located at the university, and is connected to the university's LAN and to the Internet. Dialup access to the machine, named "mona" (for "MUUG Online Network Access") is through the modem pool on the university's UMnet (Develnet). MUUG is responsible for the operation and maintenance of the machine.

The purpose of the *MUUG Online* project is to give our members access to Internet services like e-mail, ftp, and Usenet news. Any MUUG member can apply for an account on MONA to get interactive access to the Internet. We will also be providing dialup UUCP links to members who want to receive e-mail and news on their own systems instead having to manually call MONA.

Services
E-mail

Electronic mail is a very effective means of communication. It allows anyone on a networked computer, from PCs to mainframes, to send, receive, archive, forward, and distribute messages, or computer data of any kind. Many organizations have been using e-mail internally for some years. With *MUUG Online*, you will be able to either log in to MONA interactively to send mail anywhere in the world, or establish a UUCP link and integrate your present office system with the world-wide access offered by the Internet.

Everyone with an account on MONA has an e-mail address starting with their userid, like jsmith@muug.mb.ca, as well as one containing their full name, like john.smith@muug.mb.ca. Those with a UUCP connection will also have an address containing the name of their own computer, like john@JJConsult.muug.mb.ca. J&J Consulting may have users other than John, so there may also be an address like joe@JJConsult.muug.mb.ca, even though Joe doesn't have an interactive MONA account.

Usenet News

Usenet news is like a huge, world-wide, non-interactive bulletin board. Anyone with access can read and post items, and read everything everyone else has posted. The news is divided into newsgroups, each with a different topic. Topics range from politics, to humour, to computers (lots of computers...), to specialized scientific research areas.

MUUG Online intends to carry most of the newsgroups our members are interested in, taking into account disk space and communications limitations. We have also set up some local newsgroups to allow the same sort of interchange among MUUG members. News will be available interactively on MONA and via UUCP.

THIS MONTH'S MEETING

Meeting Location:
Our June meeting is scheduled for Tuesday, June 2, at 6:00 PM (a week and 1.5 hour earlier than usual). This meeting will be the traditional TUUG June BBQ. This year, Roland Schneider is hosting it at his home in Selkirk. A map is included in this month's newsletter.

Meeting Agenda:
Eat, drink, and be merry – but no computer talk!

INSIDE THIS ISSUE

UNIX is a registered trademark of UNIX System Laboratories, Inc. 1
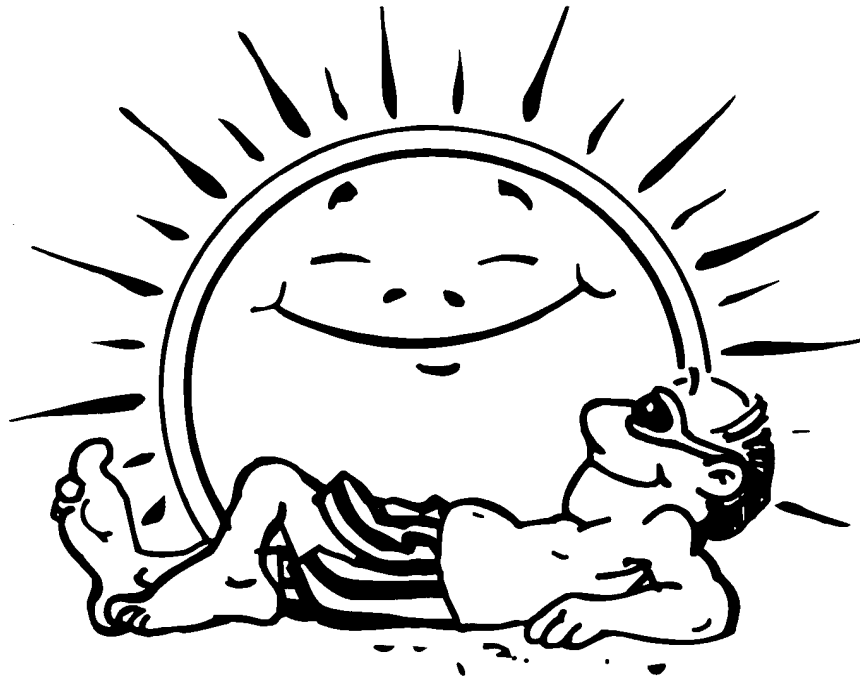
The 1991-1992 Executive

President:	Susan Zuk	(W) 788-7312
Past President:	Eric Carsted	1-883-2570
Vice-President:	Richard Kwiatkowski	589-4857
Treasurer:	Rick Horocholyn	(W) 474-4533
Secretary:	Roland Schneider	1-482-5173
Membership Sec.:	Allan Moulding	269-8054
Mailing List:	Gilles Detillieux	489-7016
Meeting Coordinator:	Kathy Norman	(W) 474-8311
Newsletter editor:	Gilbert Detillieux	489-7016
Information:	Susan Zuk	(W) 788-7312
		(FAX) 788-7450
(or)	Gilbert Detillieux	(H) 489-7016
		(FAX) 269-9178

Copyright Policy and Disclaimer

This newsletter is ©opyrighted by the Manitoba UNIX User Group. Articles may be reprinted without permission, for non-profit use, as long as the article is reprinted in its entirety and both the original author and the Manitoba UNIX User Group are given credit.

The Manitoba UNIX User Group, the editor, and contributors of this newsletter do not assume any liability for any damages that may occur as a result of information published in this newsletter.

Our Address

**Manitoba UNIX User Group
P.O. Box 130
Saint-Boniface, Manitoba
R2H 3B4**

**Internet E-mail:
editor@muug.mb.ca**

Group Information

The Manitoba UNIX User Group meets at 7:30 PM the second Tuesday of every month, except July and August. The newsletter is mailed to all paid up members one week prior to the meeting. Membership dues are $20 annually and are due at the October meeting. Membership dues are accepted by mail and dues for new members will be pro-rated accordingly.

As the Summer Approaches

By Susan Zuk, President

Much has happened in the last month in terms of group activity. The name Manitoba UNIX User Group is officially ours, we have officially become an affiliate of UniForum Canada, MUUG was visited by the President of UniForum Canada, Tom Vassos, we participated in the Muddy Waters Computer Society Computer Show, and the process of setting up internet access for our members is moving ahead quickly. So you can see this is a very exciting and busy time for the group.

The exposure we received at the Computer Show, on April 26th at the Convention Centre was excellent. Just under 4,000 people passed through the doors and we had over 400 stop and talk to us about our group and our activities. This was MUUG's first publicity event and was very worthwhile. Thanks to Gilbert Detillieux, Gilles Detillieux, Allan Moulding, Roland Schneider, and Rick Horocholyn for spending one of their precious Sundays helping to promote the group. Your dedication is wonderful.

The affiliate form was completed and sent back to the UniForum office with Tom Vassos this previous week. The first event where our group is represented is at UniForum's Open Systems Show held May 27-29. UniForum is holding one of its semi-annual meetings on Friday. This is a full day session called a National Council Meeting. The National Council is comprised of the President's of the local affiliates. I will report on the day's events in the next newsletter.

Our meeting with Tom Vassos, the President of UniForum Canada, was very enlightening. Tom spoke to the executive about various programs being pursued by the National Committee. UniForum Canada has changed its by-line to UniForum Canada, the Canadian Association of Open Systems Professionals. The feeling is that this will allow the organization to offer a fuller range of information to its members since UNIX is not the only thing which makes an environment open. Such items as standards, other operating systems and hardware can also be addressed.

Tom informed us that a Western Road Show is being planned for Edmonton, Calgary and Vancouver in the fall timeframe. This is something we could consider becoming involved with in the upcoming years. UniForum Canada is also working on developing a package price for UNIX publications for its members. They are busy talking to the publishers of the various publications. Another program is to setup a Canadian User Alliance, so that Canadian companies and users may be able to have a voice and some involvement in what is going on in the Opens Systems environment. An alliance has already been organized in the U.S. As well, industry sectors such as the Petroleum companies have organized their own groups to lobby industry to comply to their specifications. There were more items which were discussed but I will wait to receive more information and report to you in the next newsletter.

Just a reminder about our Annual Barbecue on June 9th. Come down and join us. You will find more details later in the newsletter. If you are unable to attend the Barbecue, I would like to wish you a wonderful summer and look forward to seeing you again in the fall. ➨

Manitoba UNIX User Group
Interim Financial Statements
By Rick Horocholyn, Treasurer

Balance Sheet
As of 1992/05/31

Assets	
Cash	80.00
Chequing	1454.28
Investments	8000.00
Accounts Receivable	30.00
Equipment	168.68
Other Assets	54.72
Total Assets	**9787.68**
Liabilities	
Accounts Payable	662.34
Total Liabilities	662.34
Equity	
Net income to date	8614.01
Retained Earnings (previous year)	511.33
Total Equity	9125.34
Total Liabilities and Equity	**9787.68**

Profit and Loss Statement
For the 7 months ending 1992/05/31

Revenues	
Membership Fees	1526.00
Other Revenue (Fall Symposium)	8000.00
Total Revenues	9526.00
Expenses	
Newsletter (paper & postage)	492.15
Advertising	63.00
Entertainment (Symposium wind-up)	59.55
Speaker Fees & Expenses	82.08
Bank Charges	4.83
Postal Box Rental	80.25
Miscellaneous Expenses	130.13
Total Expenses	911.99
Net Income	**8614.01**

Rick Horocholyn works for Manitoba Hydro. He's been a member of the group for several years, and has been the group's treasurer since October, 1991.

3

Using the XView OPEN LOOK Toolkit
An easy way to build a modern Graphical User Interface
By Roland Schneider

Most modern applications demand easy-to-use and easy-to-learn graphical user interfaces. (GUI's) Easy-to-use does not imply easy-to-program, of course. Toolkits, like Sun's XView, make the task more manageable, and encourage adherence to GUI standards, which in turn makes the application easier to learn because many aspects of the interface behavior will already be familiar to the user.

It is important to get a few terms straight before diving into the technical details. The underlying graphics standard is "X Windows", or "X11" or just "X". X, through the functions in the library Xlib, lets you create windows, draw lines and polygons, and monitor keyboard and mouse activity on local or networked X display servers. (display devices like workstations or X terminals) A GUI standard, like OPEN LOOK or Motif, specifies how the user interacts with a program. The functions of the mouse buttons, the appearance and operation of scrollbars, control buttons, menus, and pop-up windows are all part of the standard. A toolkit, like XView, is a set of functions which help in implementing a GUI conforming to a standard, in this case OPEN LOOK. The term "application program interface" or "API", refers to this set of functions and the arguments you pass to them.

The Details

There are very few functions to call in XView, but most take a NULL-terminated, variable-length list of arguments. The functions are easy to remember, the arguments are not. The nice thing is that most parameters have default values, so you can usually get away with only a few arguments. Functions which create something return a *handle* to the created object, which can later be used to control and query the object.

The first step is to initialize XView and create a base window into which everything else will go.

```
xv_init(NULL);
base_frame = (Frame) xv_create(NULL,
    FRAME,
    FRAME_LABEL, "My Window",
    NULL);
```

The arguments to xv_create() specify that the parent is the root window (the background), that we want to create a frame, and that its label should be "My Window." We could also have specified an icon to use when the window is closed, an initial size, whether or not the frame can be resized with the mouse, footers in addition to the header label, etc. We didn't, so XView assigns default values.

Now we have to create a panel in the frame onto which we'll put our buttons and so on.

```
control_panel = (Panel) xv_create(base_frame,
    PANEL,
    PANEL_LAYOUT, PANEL_VERTICAL,
    NULL);
```

The parent of the panel is the base frame and the layout of the items on the panel will be vertical. Now let's put a simple button on the panel.

```
xv_create(control_panel,
    PANEL_BUTTON,
    PANEL_LABEL_STRING, "Quit",
    PANEL_NOTIFY_PROC, proc_do_quit,
    NULL);
```

The parent is the control panel we created before, the thing being created is a button, the label is "Quit", and the function to call when the button is pushed is specified by the function pointer proc_do_quit.

Inverted Program Structure

"Wait a minute, that isn't how I handle user input. I read a command, parse it, and then act on it — what's this stuff about XView calling my function?" Well, that's how XView wants you to do it. It actually works quite well, since it encourages you to design your program so that it reacts to user input, or is "event driven." Unfortunately, converting an existing program to this structure can be very painful, depending on how it's currently written. There are ways to use XView with a traditional command-reading loop, but it's difficult, and many things don't happen as automatically as they do when you use the default structure.

Let's finish our program. We'll add a slider, and a non-exclusive choice item.

```
bcontrol = (Panel_item)xv_create(control_panel,
    PANEL_SLIDER,
    PANEL_LABEL_STRING, "Bright:",
    PANEL_MIN_VALUE, 0,
    PANEL_MAX_VALUE, 100,
    PANEL_SLIDER_WIDTH, 50,
    PANEL_TICKS, 5, PANEL_SHOW_VALUE, FALSE,
    PANEL_VALUE, 70,
    PANEL_NOTIFY_PROC, proc_set_brightness,
    NULL);

xv_create(control_panel, PANEL_TOGGLE,
    PANEL_LABEL_STRING, "What:",
    PANEL_CHOICE_STRINGS,
        "Inside",
        "Outside",
        NULL,
    PANEL_VALUE, 1,
    PANEL_NOTIFY_PROC, proc_set_what,
    NULL);
```

Notice that the label is always set by PANEL_LABEL_STRING, and the notification function by PANEL_NOTIFY_PROC, no matter what kind of panel item is being created. The slider has lots of parameters because the

defaults didn't suit our purpose. `PANEL_CHOICE_STRINGS` is followed by its own NULL-terminated list of labels for the choices. You can specify pictures instead of text for labels by using `PANEL_LABEL_IMAGE` instead of `PANEL_LABEL_STRING`. Similarly, `PANEL_CHOICE_STRINGS` can be replaced by `PANEL_CHOICE_IMAGES`. The `PANEL_VALUE` parameter specifies an initial value for the choice item.

The sizes of the frame and the panel have never been specified. We really just want them big enough to contain whatever is inside, so we call

```
window_fit(control_panel);/* buttons, etc. */
window_fit(base_frame);   /* contains panel */
```

Drawing Graphics

Graphics are drawn into a *canvas*, so we create one and get the corresponding X window for drawing into using Xlib calls, which won't be discussed here.

```
my_canvas = (Canvas) xv_create(base_frame,
    CANVAS,
    XV_WIDTH, 200,
    XV_HEIGHT, 100,
    NULL);


my_xwin=xv_get(canvas_paint_window(my_canvas),
    XV_XID);
```

It's also possible to set an input handling function for the canvas which will be called with the mouse coordinates, mouse buttons, keyboard keys, etc. whenever an event happens while the cursor is over the canvas.

Finishing It Off

We added the canvas to the frame, so the frame has to be resized using

```
window_fit(base_frame);   /* panel & canvas */
```

This will also stretch the panel so that it fills the entire width of the frame, which is now wider because of the canvas.

Now, finally, we give control to XView so that it can handle the input events from the mouse and keyboard, provide feedback like inverting the image of a panel button when the left mouse button is pressed, and call our notify procedure when the left mouse button is released.

```
window_main_loop(base_frame);
```

It's possible to avoid most of this programming by using software which lets you use the mouse to set panel item properties and then place them. Sun has such a product, but I've never used it, so I can't say much about it. Although this user-interface code tends to be long-winded, using it amounts to more cutting and pasting than anything else, and the compiled code isn't very large.

Notify Procedures

The processing in the program is performed by the notify procedures. Their functions can range from very simple, like the one for the quit button, to complicated and time-consuming. The parameters they are called with depend on the type of panel item being processed. Here are procedures for the panel items we created:

```
int do_inside = 1;
int do_outside = 0;
void proc_do_quit(button, event)
    Panel_item button;
    Event *event;
{

    printf("Quit button was pressed\n");
    exit(0);

}


void proc_set_brightness(item, value, event)
    Panel_item item;
    int value;
    Event *event;
{

    printf("Slider set to %d\n", value);
    /* do whatever control is supposed to do */
}


void proc_set_what(item, value, event)
    Panel_item item;
    int value;
    Event *event;
{    /* each bit reps one of the choices */
    if (value & 1) do_inside = 1;
        else do_inside = 0;
    if (value & 2) do_outside = 1;
        else do_outside = 0;
    if (value == 0)   /* nothing set */
        xv_set(bcontrol,
            PANEL_INACTIVE, TRUE,
            NULL);
    else xv_set(bcontrol,
            PANEL_INACTIVE, FALSE,
            NULL);
}
```
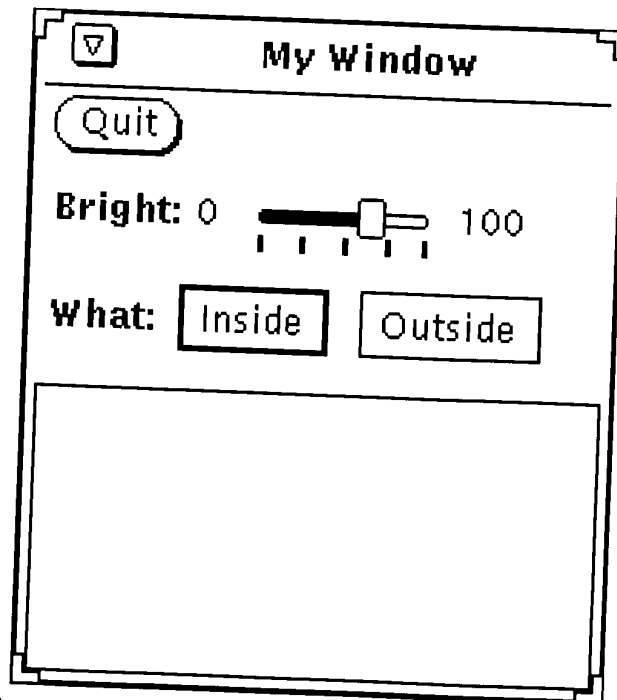
Notify procedures are always called with a handle to the panel item which generated the event. In this example, the handle was never needed because each notify procedure handles only a single panel item, something which is impractical in a large program. Notice that `proc_set_what()` uses `xv_set()` to deactivate and gray-out the brightness control if nothing is selected. In fact, `xv_set()` can change most of the parameters which can be specified to `xv_create()`, including the item labels, choice strings, values, etc.

Other Panel Items
This is only a small sample of the panel items available. There are may kinds of choice items, including abbreviated choices which automatically generate a pull-down menu. There are also text items for typing into, buttons with menus attached, message and gauge items for showing display-only information, and scrolling lists.

Other Objects
There are other kinds of XView objects too. Menus can be created and attached to panel buttons so that they will be shown when the right mouse button is pressed over the button. Alternatively, you can display a menu whenever you want, for example from a canvas input handler in response to a right mouse button press over the canvas.

One of the unique features of an OPEN LOOK menu is that it can be *pinned* by the user, making it into a small window which doesn't go away when its functions are executed. This has implications for the design of the user interface, because it allows obscure functions which may be heavily used from time to time to be buried deep in the menu structure. The user only has to "walk" through the menus once, and pin the one containing the required commands.

There are also text subwindows, which allow simple text editing, tty subwindows for running programs which want terminal I/O, scrollbars which allow the user to pan and split canvases and notices for displaying warnings and error messages. XView also provides access to selection services for cutting and pasting between applications as well as control of fonts, colormaps, cursors, and most other aspects of a modern interface.

Special Features
A serious drawback of the inverted XView programming style is that you can end up with a large number of global variables, like the handle `bcontrol`, and the status variables `do_inside` and `do_outside`, and a lot of very simple notify procedures. XView contains two useful, although somewhat subtle, features which help avoid this.

The first is a pair of functions which can be used to "walk through" all the items on a panel, avoiding the need to use a global variable for each panel item if the requirement is simply to set or read the current values. No notify procedure is needed in this case, because the values can be read with the `xv_get()` function.

The second feature is the ability to attach arbitrary data, usually in the form or pointers or integers, to any XView object, including panel and menu items. The call
```
xv_set(item,
    XV_KEY_DATA, 123, "This is my item",
    NULL);
```
attaches the key 123 and a pointer to the string "This is my item" to the item. Later, the call
```
txt = (char *)xv_get(item, XV_KEY_DATA, 123);
```
will retrieve the text. Any number of keyed data items can be attached to each XView object. This can be a useful way to communicate information from one part of a program to another and allows you to write "generic" notify procedures which can handle events from a variety of related panel items, because the items can be labeled with character string identifiers or handles to the objects they are supposed to control.

Compatibility
Theoretically, XView programs will work with any X server, but special fonts, which are now included in most systems, are required for the buttons, menus, etc. to look right. Also, some features may not be available if you don't use an OPEN LOOK compliant window manager. The window manager is the program which allows you to open, close, resize, and move windows around on the screen, and also provides the "pinning" function described earlier.

How to Get It
If you have a Sun workstation running OpenWindows, you already have XView. If you don't, source is available in both the X11R4 and X11R5 distributions from MIT. I believe the R4 version of XView had only been ported to Sun and DEC 3100 workstations; I don't know about the R5 version. X11R5 is available from MIT for the cost of distribution, and will be available for free to MUUG members when we put together our public domain software tape.

To Read Further
The standard books on X windows are published by O'Reilly and Associates, Sebastopol CA. Of particular interest are:

Volumes 1 and 2:
Xlib Programming Manual
Xlib Reference Manual
ISBN 0-937175-13-7 (set of both books)

Volume 7:
XView Programming Manual
ISBN 0-937175-52-8

Prices are about $37 per book. All three are included in Sun's on-line AnswerBook documentation. ➠

Roland Schneider is a Ph.D. student in Electrical Engineering at the University of Manitoba. He is also the MUUG secretary since October, 1991.

Using Shared Memory for Inter-Process Communication

Part 2 of 3

By Peter Graham

Last time we talked about the potential uses for shared memory both in uniprocessor and multiprocessor Unix machines. (Admittedly with a leaning towards multiprocessors. :-) This article discusses the "overhead" associated with shared memory applications; namely "synchronization."

Whenever multiple concurrent (or pseudo-concurrent) processes share data there is the possibility of contention. This simply means that more than one process may be attempting to access the data at the same time. If both of these processes read the data, then everything is alright, but if one or more attempt to write it then problems occur. Synchronization is the solution to these contention problems. By coding *semaphore* calls (or perhaps using other synchronization primitives) the programmer "synchronizes" the access of the processes to the shared data. Effectively enforcing an access order on them.

Why do we really need synchronization? This is most easily answered with a simple example of what can happen if you do not provide *synchronization*. Consider a system of automatic teller machines (ATMs). Suppose that a certain Computer Science PhD student and his wife both have ATM cards which allow access to their account. If these malicious individuals both attempt to withdraw $500 from their account at the same time (presumably using two side-by-side machines) then the bank's computer might receive the following two transactions simultaneously.

Transaction 1	Transaction 2
read balance from acct	read balance from acct
if balance < $500 then	if balance < $500 then
reject transaction	reject transaction
else	else
balance=balance-500	balance=balance-500
endif	endif
write balance to acct	write balance to acct
dispense $500	dispense $500

If we further suppose that these two identical transactions are being performed on a multiprocessor and are executed concurrently then what will happen? If the original balance was $501 (graduate students are poor :^>) then both transactions running on different processors will read the balance and find it to be $501. Since this is greater than $500, both will dispense $500 after subtracting 500 from the balance and writing the new ($1) balance back. The ending balance is correct, but $1000 has been dispensed. This is because the transactions (i.e. processes) were not synchronized. Clearly, a bank manager would not be pleased with this occurrence.

Now, before all you married folk out there go rushing out to the ATMs at your local mall, I can pretty much guarantee that the banks have closed this little loophole. This is the most common form of synchronization problem there is and it is well understood by pretty much anyone who has taken an Operating Systems or Database class.

The important thing to understand is why the problem occurs. The code reads, modifies, and then writes the account balance. The problem arises because once one transaction has read the value, the other should not be able to read it until the first has finished writing the new balance. If the second transaction can read the same balance, they will both come to the conclusion that there is $500 available in the account and will both dispense $500. Such a piece of code is called a *critical section* and access to critical sections (CSs) must be synchronized. Stated in more general terms, operations on shared data must be done *indivisably* (one at a time to completion). By synchronizing the transactions we force one to go before the other and say that they are executed *mutually exclusively*.

Now that we have that out of the way, and assuming you still want to use shared memory due to its high-efficiency, we are left with the task of learning to do synchronization using Unix. Unix provides a very flexible set of semaphore primitives for mutual exclusion (mutex). While semaphores are considered to be a low-level mutex facility, they are very general and are programming language independent. This was likely why they were chosen for use in Unix.

A semaphore is a special type of variable which is used for synchronization. Two operations are normally defined on semaphores; "wait" and "signal." The semaphore maintains an integer value which is typically either zero or one. The "wait" operation stops a given process from executing until the value of the semaphore is non-zero. It then decrements the value and returns. If a semaphore's value is zero, the "signal" operation restarts one of the processes waiting on that semaphore. Otherwise, "signal" just increments the value of the semaphore. The key here is that these are operations done in the kernel which makes them indivisible. That is, the kernel ensures that only one process can access a semaphore's value at a time. If this were not the case then we would have critical sections for accessing and updating the semaphore's value.

We can use these simple ("binary") semaphores to solve our previous synchronization problem. If we refer to the entire transaction (our CS) as 'Ti' then we simply code the following:

Transaction 1	Transaction 2
wait(mutex_sema)	wait(mutex_sema)
Ti	Ti
signal(mutex_sema)	signal(mutex_sema)

Assuming that the semaphore ('mutex_sema') is initialized to one then this will force the two transactions to be synchronized. Even if both transactions execute their 'wait' operations simultaneously, the kernel will ensure that only one is performed at a time. Thus, one transaction will get to enter its critical section while the other will be stopped. When the

transaction which first enters the CS issues its 'signal' the other process will be awakened and only then will it be allowed to enter its CS.

It is also possible to have "counting" semaphores which allow the semaphore to take on other positive values besides one and zero. The descriptions of wait and signal given above still hold in this case. We will postpone discussing their use for the time being but merely acknowledge their existence and assume their usefullness.

Unix extends the notion of the basic binary semaphore described previously by allowing counting semaphores and by also allowing semaphores to be grouped into "semaphore sets." Unix's semaphore operations operate on semaphore sets not individual semaphores (of course a set may consist of a single semaphore). The advantage of having semaphore sets is that by grouping logically related semaphores, we can perform operations on *all* members of the set at once while still being ensured of mutual exclusion.

The Unix semaphore calls consist of 'semget' to create a semaphore set, 'semop' to perform operations on a semaphore set, and 'semctl' to perform a variety of functions including destroying unused semaphores. We will discuss each in turn briefly as we did in part one of this article leaving out the details of the more esoteric features each provides.

If we require a single semaphore, the first thing we must do is create it. This can be done with the call:
```
semid=semget(key, 1, IPC_CREAT);
```
A call to 'semget', like a call to 'shmget' requires a unique key (formed using 'ftok') to identify the semaphore set. The second argument specifies the number of semaphores required (in the set) and the final argument specifies that we want to create this semaphore set if it does not already exist. As you might expect, 'semget' returns a semaphore identifier for use with the 'semop' operations or -1 if an error occurs.

A 'semop' call has the following form:
```
semop(int semid, struct sembuf **opsptr,
          unsigned int numops);
```
The first argument identifies the semaphore set being operated on and the second argument specifies a pointer to an array of semaphore operations. The third argument specifies the size of that array (i.e. the number of operations to be done.) The type 'struct sembuf' is defined as:
```
struct sembuf {
    ushort sem_num;   /* sem. # in the set */
    short sem_op;     /* op. to be performed */
    short sem_flg;    /* operation flags */
};
```
The 'sem_op' member specifies the operation to be done on the given semaphore in the set. It is used as follows:
1. If sem_op is positive, its value is added to the semaphore.
2. If sem_op is zero, the caller is made to wait until the semaphore's value becomes zero.
3. If sem_op is negative, the caller is made to wait until the semaphore's value becomes greater than or equal to the absolute value of sem_op.

There are a number of different possibilities for the 'sem-flg' member. The most useful of these permit non-blocking waits, and a roll-back facility which will correct for any outstanding semaphore operations when a process terminates abnormally. The 'semop' call returns either 0 or -1 indicating success or failure.

The 'semctl' call can be used for many purposes including explicit setting or reading of a semaphore's value. We will only consider its use in removing a semaphore which is no longer required. This can be done using the following call assuming our single semaphore example:
```
semctl(semid,0,IPC_RMID,0);
```
See the online man pages for more details about these primitive semaphore operations.

Now that we have the necessary primitives, let's look at some code which uses both shared memory and semaphores. We will implement a much simplified version of a bank's ATM system. We create a banker (server) process which initializes a set of accounts in shared memory which are then available to a collection of ATM processes. Admittedly, this is not the way an ATM system would actually be implemented but it serves its purpose for illustrating simple semaphore usage. The code in 'bank.c' creates the segment and initializes all ten accounts so they have $501.00 each. The code in 'forker.c' creates ten processes running the code in 'atm.c'. All that each ATM process does is perform the transaction described at the beginning of this article. Even numbered ATM processes operate on the account numbered with their "process number" while odd numbered processes operate on the same account as the preceding even numbered process. Thus, odd numbered accounts will not be touched while even ones will have to $500 debit transactions performed against them.

This is the reorganized output from a run without synchronization. (That is, I removed all the semaphore code.) Notice that $1000 dollars was incorrectly dispensed from all even numbered accounts.
```
account#0:initial balance is 501.000000.
account#1:initial balance is 501.000000.
account#2:initial balance is 501.000000.
account#3:initial balance is 501.000000.
account#4:initial balance is 501.000000.
account#5:initial balance is 501.000000.
account#6:initial balance is 501.000000.
account#7:initial balance is 501.000000.
account#8:initial balance is 501.000000.
account#9:initial balance is 501.000000.
acct #0:$500.00 dispensed. Please bank here again!
acct #0:$500.00 dispensed. Please bank here again!
acct #2:$500.00 dispensed. Please bank here again!
acct #2:$500.00 dispensed. Please bank here again!
acct #4:$500.00 dispensed. Please bank here again!
acct #4:$500.00 dispensed. Please bank here again!
acct #6:$500.00 dispensed. Please bank here again!
acct #6:$500.00 dispensed. Please bank here again!
acct #8:$500.00 dispensed. Please bank here again!
acct #8:$500.00 dispensed. Please bank here again!
account#0:final balance is 1.000000.
account#1:final balance is 501.000000.
```

8

account#2:final balance is 1.000000.
account#3:final balance is 501.000000.
account#4:final balance is 1.000000.
account#5:final balance is 501.000000.
account#6:final balance is 1.000000.
account#7:final balance is 501.000000.
account#8:final balance is 1.000000.
account#9:final balance is 501.000000.

This is the reorganized output from a run with synchronization. Notice that the system now performs correctly since all transactions are synchronized.

account#0:initial balance is 501.000000.
account#1:initial balance is 501.000000.
account#2:initial balance is 501.000000.
account#3:initial balance is 501.000000.
account#4:initial balance is 501.000000.
account#5:initial balance is 501.000000.
account#6:initial balance is 501.000000.
account#7:initial balance is 501.000000.
account#8:initial balance is 501.000000.
account#9:initial balance is 501.000000.
acct #0:$500.00 dispensed. Please bank here again!
acct #0: Sorry not enough money.
acct #2:$500.00 dispensed. Please bank here again!
acct #2: Sorry not enough money.
acct #4:$500.00 dispensed. Please bank here again!

acct #4: Sorry not enough money.
acct #6:$500.00 dispensed. Please bank here again!
acct #6: Sorry not enough money.
acct #8:$500.00 dispensed. Please bank here again!
acct #8: Sorry not enough money.
account#0:final balance is 1.000000.
account#1:final balance is 501.000000.
account#2:final balance is 1.000000.
account#3:final balance is 501.000000.
account#4:final balance is 1.000000.
account#5:final balance is 501.000000.
account#6:final balance is 1.000000.
account#7:final balance is 501.000000.
account#8:final balance is 1.000000.
account#9:final balance is 501.000000.

Notice that the code does not remove the semaphore at any point. This is because we cannot guarantee an ordering on the execution of the ATM processes so none of them can destroy it for fear of doing so before all the other processes were finished. The semaphore could have been created by the banker process and explicitly removed after the 60 seconds. In the interest of brevity I simply omitted this code.

Next time we will look at the development of a print spooler application which uses shared memory and semaphores, and the resulting code. ➡

```
/*********/
/* acct.h */
/*********/
/* balances for 10 accounts - small bank */
typedef struct acct {
        float balances[10];
} ACCT, *ACCTPTR;


/***********/
/* forker.c */
/***********/

main()
{
        int    i,          /* simple counter      */
               pid;        /* forked process' pid */
        char   argstr[16]; /* converted pid string */

        /* this code simply creates ten identical */
        /* processes to run the program 'atm.c'   */
        /* which simulates the actions of an      */
        /* Automated Teller Machine (ATM).        */
        for (i=0;i<10;i++) {
                pid=fork(); /* fork a new process   */
                if (pid==0) {
                        /* we are child process so... */
                        /* execute the atm program,   */
                        /* passing it a process num   */
                        sprintf(argstr,"%d",i);
                        execl("/home/cs/staff/pgraham/
misc/tuug/shm_article/part2/atm",
                             "atm",argstr,(char *)0);
                }
        }
} /* end main */
```

```
/*********/
/* bank.c */
/*********/

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include "acct.h"

/* necessary because of problem   */
/* with SunOS include file shm.h */
#define SHM_W 0200      /* shm write permission */
#define SHM_R 0400      /* shm read permission  */

main()
{
        int        i;          /* simple counter */
        key_t      acct_shmkey; /* key to shm seg*/
        int        acct_segid; /* shm segment ID */
        ACCTPTR    acct_segaddr;      /* seg ptr  */
        /* need for call to 'shmctl()':        */
        struct shmid_ds   *acct_shmbfptr;

        /* this code creates the shared memory   */
        /* segment and initializes it.           */

        /* creating unique key/name for acct seg */
        if ((acct_shmkey=ftok("/home/cs/staff/
pgraham/misc/tuug/shm_article/part2/forker.c",
                                       'M'))==-1) {
            printf(
            "Couldn't create shared memory key.\n");
            exit(-1);
        }
```

9

```
/* call shmget to create it              */
if ((acct_segid=shmget(acct_shmkey,
          sizeof(ACCT),
          IPC_CREAT|SHM_R|SHM_W))==-1) {
     printf(
"Couldn't get the shared memory segment.\n");
     exit(-1);
}

/* and map it into our address space at   */
/* address returned in 'acct_segaddr'.    */
if ((acct_segaddr=(ACCTPTR)
          shmat(acct_segid,(char *)0,0))
                    ==(ACCTPTR) (-1)) {
     printf(
  "Couldn't attach shared memory segment.\n");
     exit(-1);
}

/* Put some balances in the accounts      */
for (i=0;i<10;i++) {
     acct_segaddr->balances[i]=501.00;
}
printf(
"Bank is open for business for 1 minute.\n");
/* allow time to run the ATM processes    */
sleep(60);

/* detach the shared memory segment        */
if (shmdt((char *) acct_segaddr)==-1) {
     printf(
  "Couldn't detach shared memory segment.\n");
}

/* now remove shared segment altogether    */
if (shmctl(acct_segid,IPC_RMID,
              acct_shmbfptr)==-1) {
     printf(
  "Couldn't remove shared memory segment.\n");
}
}

/********/
/* atm.c */
/********/

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include "acct.h"

/* necessary because of problem  */
/* with SunOS include file shm.h */
#define SHM_W 0200     /* shm write permission */
#define SHM_R 0400     /* shm read permission  */

/* Semaphore wait operation */
s_wait(sema)
int    sema;
{
     static struct sembuf op_wait[2] = {
```

```
          0, 0, 0,    /* wait for zero semaph.*/
          0, 1, 0     /* then increment by 1  */
     };

     semop(sema, &op_wait[0], 2);
} /* s_wait */

/* Semaphore signal operation */
s_signal(sema)
int    sema;
{
     static struct sembuf op_signal[1] = {
          0, -1, IPC_NOWAIT,
          /* decrement semaphore by 1      */
          /* NOWAIT ensures no waiting      */
     };

     semop(sema, &op_signal[0], 1);
} /* s_signal */


transaction(acct,acct_segaddr)
int        acct; /* account number to work on */
ACCTPTR    acct_segaddr; /* mapped segment ptr */
{
     float balance;    /* current balance      */

     /* read the balance */
     balance=acct_segaddr->balances[acct];

     /* sleep so as to force a context switch */
     /* to simulate concurrent execution of    */
     /* the processes.                         */
     sleep(1);

     /* check the balance */
     if (balance < 500.00) {
          printf(
  "acct #%d: Sorry not enough money.\n",acct);
     } else {
          /* debit the balance */
          balance-=500.00;

          /* write the balance back */
          acct_segaddr->balances[acct]=balance;

          /* dispense the money */
          printf("acct #%d:%s\n",
  "$500.00 dispensed. Please bank here again!",
               acct);
     }
} /* transaction */


main(argc,argv)
int    argc;
char   *argv[];
{
     int    i,             /* simple counter       */
            process_num; /* this process' num    */
     key_t acct_shmkey,   /* shm segment key      */
            acct_semkey; /* shm semaphore key    */
```

10

```
int   acct_segid,  /* shm segment ID    */
      acct_semid;  /* shm semaphore ID  */
/* pointer to mapped shared segment:    */
ACCTPTR    acct_segaddr;

/* this code runs some sample          */
/* transactions against the accounts   */

/* create unique key/name for shared seg */
if ((acct_shmkey=ftok("/home/cs/staff/
pgraham/misc/tuug/shm_article/part2/forker.c",
                            'M'))==-1) {
      printf(
      "Couldn't create shared memory key.\n");
      exit(-1);
}

/* call shmget to "open" it */
if ((acct_segid=shmget(acct_shmkey,
            sizeof(ACCT),SHM_R|SHM_W))
                            ==-1) {
      printf(
"Couldn't get the shared memory segment.\n");
      exit(-1);
}

/* map it into our address space at an   */
/* address returned in 'acct_segaddr'.   */
if ((acct_segaddr=(ACCTPTR)
            shmat(acct_segid,(char *)0,0))
                  ==(ACCTPTR)(-1)) {
      printf(
 "Couldn't attach shared memory segment.\n");
      exit(-1);
}

/* create unique key/name for semaphore  */
if ((acct_semkey=ftok("/home/cs/staff/
pgraham/misc/tuug/shm_article/part2/forker.c",
                            'S'))==-1) {
      printf("Couldn't create semaphore.\n");
      exit(-1);
}
```

```
/* call semget to get the semaphore      */
if ((acct_semid=semget(acct_semkey,1,
            IPC_CREAT|SHM_R|SHM_W))==-1) {
      printf(
            "Couldn't get the semaphore.\n");
      exit(-1);
}

/* extract our process num from argv[1]  */
sscanf(argv[1],"%d",&process_num);

/* print initial balances */
printf(
      "account#%d:initial balance is %f.\n",
      process_num,
      acct_segaddr->balances[process_num]);

/* If we are an even process, we reference*/
/* acct number specified by our process #.*/
/* If odd, we reference the previous acct.*/
/* This way we ensure the possibility of  */
/* synchronization problems.              */
if ((process_num%2)==0) { /* even numbered*/
      s_wait(acct_semid);
      transaction(process_num,acct_segaddr);
      s_signal(acct_semid);
} else {          /* odd numbered process */
      s_wait(acct_semid);
      transaction(process_num-1,
                  acct_segaddr);
      s_signal(acct_semid);
}

/* print final balances */
printf("account#%d:final balance is %f.\n",
      process_num,
      acct_segaddr->balances[process_num]);

/* detach the shared memory segment      */
if (shmdt((char *) acct_segaddr)==-1) {
      printf(
 "Couldn't detach shared memory segment.\n");
}
}
```

Annual MUUG Barbecue
June 2, 6:00 pm

Host: Roland Schneider
        phone: 1-482-5173

Where: The Schneiders'
        8483 Henderson Hwy N. (PR 204)
        Across from *The Strawberry Patch*
        About 20 minutes from the Perimeter
        (See map)

When: Tuesday, June 2, 6:00 pm
        (1 week earlier than our normal meetings)

Bring: Meat to cook
        Beer
        Lawn chairs

RSVP: Roland Schneider
        1-482-5173
        (days and evenings)
        e-mail: rsch@muug.mb.ca
        or
        Susan Zuk
        788-7312 (days)

Map shows routes: To Selkirk, Strawberry Patch, PR 509, PTH 9 (Main St.), Schneider's 8483 Henderson Hwy. N., PR 204 (Henderson N.), Lockport, Red River, PR 204 (Henderson), PTH 44, Floodway, PTH 59, To Winnipeg. Map is NOT drawn to scale.
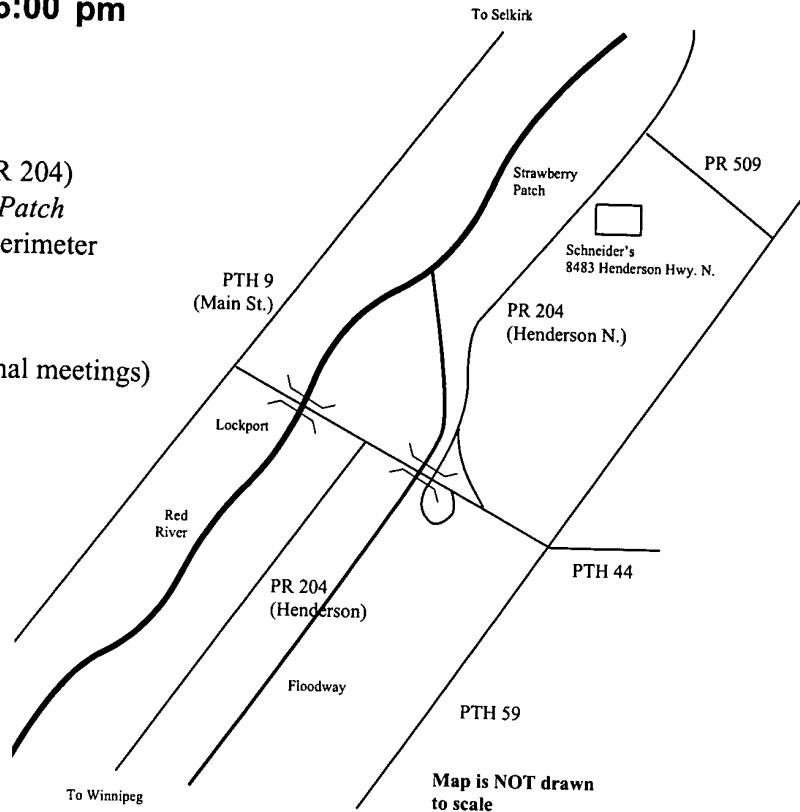
We will supply chips and other nibble food, soft drinks, salads, and a cake. (and insect repellent if necessary) Spouses or significant others, and children, are also welcome. Bring a swimsuit if you want to take a dip in our pond.

TUUG Meeting Minutes
Tuesday, May 12, 1992, 7:30 PM
234B Engineering Bldg., University of Manitoba, Ft. Garry Campus

Chair: Susan Zuk
Attendance: 47

Business meeting:
a) President's Report
- There was significant interest in MUUG's booth at the MWCS computer fest.
- The MUUG Online project is making progress.
- MUUG hopes for future joint projects with CIPS.
- The process of affiliation with UniForum is proceeeding.

b) Membership Report
- TUUG currently has 84 members

c) Treasuer's Report
- MUUG has $1800 in chequing account and $8000 invested.

d) New Business
- Moved by Gilbert Detillieux, seconded by Peter Graham, that the MUUG executive be authorized to spend up to $2600 to aquire a large SCSI disk for the MUUG Online system.
  - other, preferably free, alternatives will be explored before a disk is purchased.
  - passed unanimously.

Presented topic:
Future trends at Intel – Jon Coxworth, Intel Corp.

MUUG Goes Online!
Continued from page 1

Ftp

Ftp stands for 'File Transfer Protocol', and is used to copy ASCII and binary information across the Internet. Ftp is the usual method of obtaining PD software from other sites. It is usually used interactively to search through a restricted area on a remote site's disk and retrieve the desired data. Because ftp is an interactive program, this service will not be available through UUCP, although we may be able to set up an ftp-via-UUCP or ftp-via-e-mail service in the future.

UUCP

UUCP (UNIX to UNIX CoPy) is a system of programs which allows files, e-mail, news, and other information to be transmitted via dialup modem connections. When properly set up, all communications happen in the background, and the link is essentially transparent. *MUUG Online* will be providing UUCP links as soon as possible. To get a UUCP link, you will first need an account on MONA. Further information will be posted on MONA when we are ready to begin offering UUCP service.

PD Software

There is a lot of public domain software available for UNIX systems. Instead of having everyone retrieve the same software from the Internet repeatedly, we want to establish a repository of current versions of general-interest software on MONA. This software will then also be made available to MUUG members via tape and/or floppy disk.

Costs

There are some real costs involved in running the *MUUG Online* project. Although we don't pay anything for the Sun386i or it's network connections, we do have to fix or replace it if it breaks, and we have to obtain a large SCSI disk (1Gb) to provide spool space for news and storage for downloaded PD software. We are working on getting the disk for as little money as possible, perhaps even for free. Access to *MUUG Online* will be free for MUUG members until October, after which there will likely be a small ($30-$40 per year) charge for using the system.

Instructions

To access MONA interactively, you will need to get an account application form, fill it in, and mail it to MUUG or give it to a member of the executive. Account applications were included in last month's newsletter.

Once you have an account, use your modem to dial 275-6100 (300/1200/2400 bps) or 275-6132 (9600 bps, V.42bis, MNP5). Press RETURN once the connection is established and answer "muug" to the classname prompt. You will then be connected to MONA and asked for your userid and password. You will then be asked for your terminal type. Many communications packages emulate a DEC VT100. If you don't know what type of terminal you are emulating, "vt100" or "ansi" are good choices, with "dumb" as a last resort. (After you have logged on, look in the file /etc/termcap for a list of all the available terminal types, or type "help termtype")

Once you have logged in, you can type the command "news" to find out about the status of the MUUG Online project and other information of interest to MUUG members. Note that this is not related to the Usenet news described above. Use the command "help" to obtain advice about various procedures. Use "man" to access the online UNIX manuals.

Thanks

The entire *MUUG Online* project would not be possible without the generosity of the University of Manitoba Computer Centre and the assistance of the people there. We would especially like to thank Bill Reid, Kathy Norman, and Gary Mills. MUUG members continue to put a lot of effort into the project. Thanks are due to Andrew Chan for setting MONA up and getting all sorts of stuff working, to Gilles Detillieux for his work on the e-mail configuration, and to Gilbert Detillieux for setting up and coordinating many other aspects of the project.

To Find Out More

To find out more about *MUUG Online*, or to get a *MUUG Online* application form, please contact Roland Schneider at 1-482-5173 (days and evenings) or send e-mail to "info@muug.mb.ca". ➛

Roland Schneider is a Ph.D. student in Electrical Engineering at the University of Manitoba. He is also the MUUG secretary since October, 1991.

Coming Up

Meeting:
Our next meeting is scheduled for Tuesday, September 8, at 7:30 PM (since we don't meet in July and August). Meeting topic and location will be given in September's newsletter. Meanwhile, enjoy the summer!

Online:
Watch for changes this summer to MUUG Online – more disk space, more software. Also, we'll try to keep a round table forum going on the local news groups.

Newsletter:
We will likely continue with RPC Programming by Scott Balneaves. We will also have part 3 on shared memory by Peter Graham, and several "filler" articles by Roland Schneider. Thanks again to all those who submitted those great articles throughout the year. Please keep the articles coming this summer – we'll need lots of material for the fall. Also keep in mind that nominations for the elections come up in September.

http://www.freebsd.org
FreeBSD

Select a server near you:

| IPv6 Austria | ▾ | Go |

Language: Japanese, Spanish, Russian, Other

News
· Announcements
· In the Press
· More ...

Software
· Getting FreeBSD
· Release Information
· Ported Applications

Documentation
· For Newbies
· Handbook
· FAQ
· Doc. Project
· More...

Support
· Mailing lists
· Newsgroups
· User Groups
· Web Resources
· Security
· More...

Bug Reports
· Send a bug report
· View open reports
· Search by bug ID
· More...

Development
· Projects
· Release Engineering
· CVS Repository

Vendors
· Software
· Hardware
· Consulting
· Misc

Donations
· Donations Liaison
· Current Donations
· List of needs

What is FreeBSD?

FreeBSD is an advanced operating system for x86 compatible, DEC Alpha, IA-64, PC-98 and UltraSPARC architectures. It is derived from BSD UNIX, the version of UNIX developed at the University of California, Berkeley. It is developed and maintained by a large team of individuals. Additional platforms are in various stages of development.

Cutting edge features

FreeBSD offers advanced networking, performance, security and compatibility features today which are still missing in other operating systems, even some of the best commercial ones.

Powerful Internet solutions

FreeBSD makes an ideal Internet or Intranet server. It provides robust network services under the heaviest loads and uses memory efficiently to maintain good response times for thousands of simultaneous user processes. Visit our gallery for examples of FreeBSD powered applications and services.

Run a huge number of applications

The quality of FreeBSD combined with today's low-cost, high-speed PC hardware makes FreeBSD a very economical alternative to commercial UNIX workstations. It is well-suited for a great number of both desktop and server

New Technology Release: 5.0
· Announcement
· Installation Guide
· Release Notes
· Hardware Notes
· Errata
· Early Adopter's Guide

Production Release: 4.7
· Announcement
· Installation Guide
· Release Notes
· Hardware Notes
· Errata

Project News
Latest update: January 30, 2003
· New committer: Hartmut Brandt (Sparc and ATM)
· New committer: Hideyuki KURASHINA (Documentation Project)
· FreeBSD 5.0-RELEASE is now available
· New committer: Christian Brüffer (Documentation Project)
· New committer: Michael Telahun Makonnen
· September 2002 - October 2002 Status Report
· BSD Conference Japan 2002
· FreeBSD 5.0 Developer Preview #2 Now Available
· nVidia releases Geforce drivers for FreeBSD! Check out the README for more information.
· New committer: Stéphane Legrand (Documentation Project)
More...

FreeBSD Press
Latest update: February 2003
· FreeBSD 5.0 looks to the enterprise
· Odds and Ends
· FreeBSD 5.0 Unleashed
· Opera Software Releases Version for FreeBSD
· DVD Playback on FreeBSD
· The BSDs: Sophisticated, Powerful and (Mostly) Free
· Using Sound on FreeBSD

number of both desktop and server applications.

Easy to install

FreeBSD can be installed from a variety of media including CD-ROM, DVD-ROM, floppy disk, magnetic tape, an MS-DOS partition, or if you have a network connection, you can install it *directly* over anonymous FTP or NFS. All you need is a pair of blank, 1.44MB floppies and these directions.

FreeBSD is *free*

While you might expect an operating system with these features to sell for a high price, FreeBSD is available free of charge and comes with full source code. If you would like to try it out, more information is available.

· Using Sound on FreeBSD
· BSD, An Enterprise OS? Well, Yes
· Turn FreeBSD into a Multimedia Workstation
More...

Security Advisories
Latest update: February 04, 2003
· FreeBSD-SA-03:01.cvs
· FreeBSD-SA-02:44.filedesc
· FreeBSD-SA-02:43.bind
· FreeBSD-SA-02:41.smrsh
· FreeBSD-SA-02:42.resolv
· FreeBSD-SA-02:40.kadmind
· FreeBSD-SN-02:06
· FreeBSD-SA-02:39.libkvm
· FreeBSD-SN-02:05
More...

To learn more about FreeBSD, visit our gallery of FreeBSD related publications or FreeBSD in the press, and browse through this website!

Contributing to FreeBSD

It is easy to contribute to FreeBSD. All you need to do is find a part of FreeBSD which you think could be improved and make those changes (carefully and cleanly) and submit that back to the Project by means of send-pr or a committer, if you know one. This could be anything from documentation to artwork to source code. See the Contributing to FreeBSD article for more information.

Even if you're not a programmer, there are other ways to contribute to FreeBSD. The FreeBSD Foundation is a non-profit organization for which direct contributions are fully tax deductible. Please contact bod@FreeBSDFoundation.org for more information or write to: The FreeBSD Foundation, 7321 Brockway Dr. Boulder, CO 80303. USA

Silicon Breeze has also sculpted and cast

the BSD Daemon in metal and is now
donating 15% of all proceeds from these
statuettes back to the FreeBSD Foundation.
The complete story and information on how
to order a BSD Daemon is available from
this page.

Changes to the web site go live at 0800 and 2000 UTC every day.

Contact us   <u>Copyright</u> (c) 1995-2003 The FreeBSD Project.
$FreeBSD: www/en/index.xsl,v 1.53 2003/02/21 11:59:21 wosch Exp $ All rights reserved.

Frequently Asked Questions for FreeBSD 2.X, 3.X and 4.X

The FreeBSD Documentation Project

Copyright © 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003 by The FreeBSD Documentation Project

This is the FAQ for FreeBSD versions 2.X, 3.X, and 4.X. All entries are assumed to be relevant to FreeBSD 2.0.5 and later, unless otherwise noted. If you are interested in helping with this project, send email to the FreeBSD documentation project mailing list <freebsd-doc@FreeBSD.org>. The latest version of this document is always available from the FreeBSD World Wide Web server. It may also be downloaded as one large HTML file with HTTP or as plain text, PostScript, PDF, etc. from the FreeBSD FTP server. You may also want to Search the FAQ.

Table of Contents

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

For questions about FreeBSD, read the documentation before contacting <questions@FreeBSD.org>.

For questions about this documentation, e-mail <doc@FreeBSD.org>.

Frequently Asked Questions for FreeBSD 2.X, 3.X and 4.X

Chapter 1 Introduction

Welcome to the FreeBSD 2.X-4.X FAQ!

As is usual with Usenet FAQs, this document aims to cover the most frequently asked questions concerning the FreeBSD operating system (and of course answer them!). Although originally intended to reduce bandwidth and avoid the same old questions being asked over and over again, FAQs have become recognized as valuable information resources.

Every effort has been made to make this FAQ as informative as possible; if you have any suggestions as to how it may be improved, please feel free to mail them to the FreeBSD documentation project mailing list <freebsd-doc@FreeBSD.org>.

1.1. What is FreeBSD?

Briefly, FreeBSD is a UN*X-like operating system for the i386, IA-64, PC-98, Alpha/AXP, and

UltraSPARC platforms based on U.C. Berkeley's ``4.4BSD-Lite'' release, with some ``4.4BSD-Lite2'' enhancements. It is also based indirectly on William Jolitz's port of U.C. Berkeley's ``Net/2'' to the i386, known as ``386BSD'', though very little of the 386BSD code remains. A fuller description of what FreeBSD is and how it can work for you may be found on the FreeBSD home page.

FreeBSD is used by companies, Internet Service Providers, researchers, computer professionals, students and home users all over the world in their work, education and recreation. See some of them in the FreeBSD Gallery.

For more detailed information on FreeBSD, please see the FreeBSD Handbook.

1.2. What is the goal of the FreeBSD Project?

The goal of the FreeBSD Project is to provide software that may be used for any purpose and without strings attached. Many of us have a significant investment in the code (and project) and would certainly not mind a little financial compensation now and then, but we definitely do not insist on it. We believe that our first and foremost ``mission'' is to provide code to any and all comers, and for whatever purpose, so that the code gets the widest possible use and provides the widest possible benefit. This is, we believe, one of the most fundamental goals of Free Software and one that we enthusiastically support.

That code in our source tree which falls under the GNU General Public License (GPL) or GNU Library General Public License (LGPL) comes with slightly more strings attached, though at least on the side of enforced access rather than the usual opposite. Due to the additional complexities that can evolve in the commercial use of GPL software, we do, however, endeavor to replace such software with submissions under the more relaxed FreeBSD license whenever possible.

1.3. Does the FreeBSD license have any restrictions?

Yes. Those restrictions do not control how you use the code, merely how you treat the FreeBSD Project itself. If you have serious license concerns, read the actual license. For the simply curious, the license can be summarized like this.

- Do not claim that you wrote this.

- Do not sue us if it breaks.

1.4. Can FreeBSD replace my current operating system?

For most people, yes. But this question is not quite that cut-and-dried.

Most people do not actually use an operating system. They use applications. The applications are what really use the operating system. FreeBSD is designed to provide a robust and full-featured environment for applications. It supports a wide variety of web browsers, office suites, email readers, graphics programs, programming environments, network servers, and just about everything else you might want. Most of these applications can be managed through the Ports Collection.

If you need to use an application that is only available on one operating system, you simply cannot replace that operating system. Chances are there is a very similar application on FreeBSD, however. If you want a solid office or Internet server, a reliable workstation, or just the ability to do your job without interruptions, FreeBSD will almost certainly do everything you need. Many computer users across the world, including both novices and experienced UNIX® administrators, use FreeBSD as their only desktop operating system.

If you are migrating to FreeBSD from some other UNIX® environment, you already know most of what you need to. If your background is in graphic-driven operating systems such as Windows and older versions of Mac OS, expect to invest additional time learning the UNIX® way of doing things. This FAQ and the FreeBSD Handbook are excellent places to start.

1.5. Why is it called FreeBSD?

- It may be used free of charge, even by commercial users.

- Full source for the operating system is freely available, and the minimum possible restrictions have been placed upon its use, distribution and incorporation into other work (commercial or non-commercial).

- Anyone who has an improvement or bug fix is free to submit their code and have it added to the source tree (subject to one or two obvious provisions).

  It is worth pointing out that the word ``free" is being used in two ways here, one meaning ``at no cost", the other meaning ``you can do whatever you like". Apart from one or two things you *cannot* do with the FreeBSD code, for example pretending you wrote it, you can really do whatever you like with it.

1.6. What is the latest version of FreeBSD?

At this point in FreeBSD's development, there are two parallel development branches; releases are being made from both branches. The 4.X series of releases is being made from the *-STABLE* branch and the 5.X series of releases is being made from *-CURRENT*.

Version 5.0 is the latest release from the *-CURRENT* branch; it was released in January 2003. Version 4.7 is the latest release from the *-STABLE* branch; it was released in October 2002.

Briefly, *-STABLE* is aimed at the ISP, corporate user, or any user who wants stability and a minimal number of changes compared to the new (and possibly unstable) features of the latest *-CURRENT* snapshot. Releases can come from either branch, but *-CURRENT* should only be used if you are prepared for its increased volatility (relative to *-STABLE*, that is).

Releases are made every few months. While many people stay more up-to-date with the FreeBSD sources (see the questions on FreeBSD-CURRENT and FreeBSD-STABLE) than that, doing so is more of a commitment, as the sources are a moving target.

More information on FreeBSD releases can be found on the Release Engineering page on the FreeBSD Web site.

1.7. What is FreeBSD-CURRENT?

FreeBSD-CURRENT is the development version of the operating system, which will in due course become 5.0-RELEASE. As such, it is really only of interest to developers working on the system and die-hard hobbyists. See the relevant section in the handbook for details on running -CURRENT.

If you are not familiar with the operating system or are not capable of identifying the difference between a real problem and a temporary problem, you should not use FreeBSD-CURRENT. This branch sometimes evolves quite quickly and can be un-buildable for a number of days at a time. People that use FreeBSD-CURRENT are expected to be able to analyze any problems and only report them if they are deemed to be mistakes rather than ``glitches''. Questions such as ``make world produces some error about groups'' on the -CURRENT mailing list may be treated with contempt.

Every day, snapshot releases are made based on the current state of the -CURRENT and -STABLE branches. Distributions of the occasional snapshot are made available. The goals behind each snapshot release are:

- To test the latest version of the installation software.

- To give people who would like to run -CURRENT or -STABLE but who do not have the time or bandwidth to follow it on a day-to-day basis an easy way of bootstrapping it onto their systems.

- To preserve a fixed reference point for the code in question, just in case we break something really badly later. (Although CVS normally prevents anything horrible like this happening :)

- To ensure that all new features and fixes in need of testing have the greatest possible number

of potential testers.

No claims are made that any -CURRENT snapshot can be considered ``production quality'' for any purpose. If you want to run a stable and fully tested system, you will have to stick to full releases, or use the -STABLE snapshots.

Snapshot releases are directly available from ftp://current.FreeBSD.org/pub/FreeBSD/ for 5.0-CURRENT and releng4.FreeBSD.org for 4-STABLE snapshots. 3-STABLE snapshots are not being produced at the time of this writing (May 2000).

Snapshots are generated, on the average, daily for all actively developed branches.

1.8. What is the FreeBSD-STABLE concept?

Back when FreeBSD 2.0.5 was released, FreeBSD development branched in two. One branch was named -STABLE, one -CURRENT. FreeBSD-STABLE is intended for Internet Service Providers and other commercial enterprises for whom sudden shifts or experimental features are quite undesirable. It receives only well-tested bug fixes and other small incremental enhancements. FreeBSD-CURRENT, on the other hand, has been one unbroken line leading towards 5.0-RELEASE (and beyond) since 2.0 was released. If a little ASCII art would help, this is how it looks:

```
                  2.0
                   |
                   |
                   |  [2.1-STABLE]
     *BRANCH*      2.0.5 -> 2.1 -> 2.1.5 -> 2.1.6 -> 2.1.7.1   [2.1-STABLE ends]
                   |                                 (Mar 1997)
                   |
                   |
                   |  [2.2-STABLE]
     *BRANCH*      2.2.1 -> 2.2.2-RELEASE -> 2.2.5 -> 2.2.6 -> 2.2.7 -> 2.2.8 [end]
                   |            (Mar 1997)     (Oct 97) (Apr 98) (Jul 98) (Dec 98)
                   |
                   |
                  3.0-SNAPs   (started Q1 1997)
                   |
                   |
                  3.0-RELEASE (Oct 1998)
                   |
                   |  [3.0-STABLE]
     *BRANCH*      3.1-RELEASE   (Feb 1999) -> 3.2 -> 3.3 -> 3.4 -> 3.5 -> 3.5.1
                   |                            (May 1999) (Sep 1999) (Dec 1999) (June 20
                   |
```

```
        |   [4.0-STABLE]
*BRANCH*        4.0   (Mar 2000) -> 4.1 -> 4.1.1 -> 4.2 -> 4.3 -> 4.4 -> ... futu
        |
        |                   (July 2000)    (Sep 2000)    (Nov 2000)
      \ | /
        +
     [5.0-CURRENT continues]
```

The 2.2-STABLE branch was retired with the release of 2.2.8. The 3-STABLE branch has ended with the release of 3.5.1, the final 3.X release. The only changes made to either of these branches will be, for the most part, security-related bug fixes.

4-STABLE is the actively developed -STABLE branch. The latest release on the 4-STABLE is 5.0-RELEASE, which was released in January 2003.

The 5-CURRENT branch is slowly progressing toward 5.0-RELEASE and beyond. See What is FreeBSD-CURRENT? for more information on this branch.

1.9. When are FreeBSD releases made?

The Release Engineering Team <re@FreeBSD.org> releases a new version of FreeBSD about every four months, on average. Release dates are announced well in advance, so that the people working on the system know when their projects need to be finished and tested. A testing period precedes each release, in order to ensure that the addition of new features does not compromise the stability of the release. Many users regard this caution as one of the best things about FreeBSD, even though waiting for all the latest goodies to reach -STABLE can be a little frustrating.

More information on the release engineering process (including a schedule of upcoming releases) can be found on the release engineering pages on the FreeBSD Web site.

For people who need or want a little more excitement, binary snapshots are made daily as discussed above.

1.10. Who is responsible for FreeBSD?

The key decisions concerning the FreeBSD project, such as the overall direction of the project and who is allowed to add code to the source tree, are made by a core team of 9 people. There is a much larger team of more than 200 committers who are authorized to make changes directly to the FreeBSD source tree.

However, most non-trivial changes are discussed in advance in the mailing lists, and there are no

restrictions on who may take part in the discussion.

1.11. Where can I get FreeBSD?

Every significant release of FreeBSD is available via anonymous FTP from the FreeBSD FTP site:

- For the current 3.X-STABLE release, 3.5.1-RELEASE, see the 3.5.1-RELEASE directory.

- The latest 5.X release, 5.0-RELEASE can be found in the 5.0-RELEASE directory.

- The latest 4-STABLE release, 4.7-RELEASE can be found in the 4.7-RELEASE directory.

- 4.X snapshots are usually made daily.

- 5.0 Snapshot releases are made daily for the -CURRENT branch, these being of service purely to bleeding-edge testers and developers.

Information about obtaining FreeBSD on CD, DVD, and other media can be found in the Handbook.

1.12. How do I set up a FreeBSD mirror?

Information on setting up a FreeBSD mirror can be found in the Mirroring FreeBSD article.

1.13. How do I access the Problem Report database?

The Problem Report database of all user change requests may be queried (or submitted to) by using our web-based PR submission and query interfaces. The send-pr(1) command can also be used to submit problem reports and change requests via electronic mail.

Before submitting a problem report, please read Writing FreeBSD Problem Reports, an article on how to write good problem reports.

1.14. How do I become a FreeBSD Web mirror?

There are multiple ways to mirror the Web pages.

- You can retrieve the formatted files from a FreeBSD CVSup server using the application `net/cvsup`. The file `/usr/share/examples/cvsup/www-supfile` contains an example CVSup configuration file for web mirrors.

- You can download the web site source code from any FreeBSD FTP server using your favorite

ftp mirror tool. Keep in mind that you have to build these sources before publishing them. Start mirroring at ftp://ftp.FreeBSD.org/pub/FreeBSD/FreeBSD-current/www.

1.15. What other sources of information are there?

Please check the Documentation list on the main FreeBSD web site.

Prev   Home   Next
Frequently Asked Questions for   Documentation and Support
FreeBSD 2.X, 3.X and 4.X

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

For questions about FreeBSD, read the documentation before contacting <questions@FreeBSD.org>.
For questions about this documentation, e-mail <doc@FreeBSD.org>.

FreeBSD Handbook

Chapter 1 Introduction

1.3 About the FreeBSD Project

The following section provides some background information on the project, including a brief history, project goals, and the development model of the project.

1.3.1 A Brief History of FreeBSD

Contributed by Jordan Hubbard.

The FreeBSD project had its genesis in the early part of 1993, partially as an outgrowth of the ``Unofficial 386BSD Patchkit'' by the patchkit's last 3 coordinators: Nate Williams, Rod Grimes and myself.

Our original goal was to produce an intermediate snapshot of 386BSD in order to fix a number of problems with it that the patchkit mechanism just was not capable of solving. Some of you may remember the early working title for the project being ``386BSD 0.5'' or ``386BSD Interim'' in reference to that fact.

386BSD was Bill Jolitz's operating system, which had been up to that point suffering rather severely from almost a year's worth of neglect. As the patchkit swelled ever more uncomfortably with each passing day, we were in unanimous agreement that something had to be done and decided to assist Bill by providing this interim ``cleanup'' snapshot. Those plans came to a rude halt when Bill Jolitz suddenly decided to withdraw his sanction from the project without any clear indication of what would be done instead.

It did not take us long to decide that the goal remained worthwhile, even without Bill's support, and so we adopted the name ``FreeBSD'', coined by David Greenman. Our initial objectives were set after consulting with the system's current users and, once it became clear that the project was on the road to perhaps even becoming a reality, I contacted Walnut Creek CDROM with an eye toward improving FreeBSD's distribution channels for those many unfortunates without easy access to the Internet. Walnut Creek CDROM not only supported the idea of distributing FreeBSD on CD but also went so far as to provide the project with a machine to work on and a fast Internet connection. Without Walnut Creek CDROM's almost unprecedented degree of faith in what was, at the time, a completely unknown project, it is quite unlikely that FreeBSD would have gotten as far, as fast, as it has today.

The first CDROM (and general net-wide) distribution was FreeBSD 1.0, released in December of 1993. This was based on the 4.3BSD-Lite (``Net/2'') tape from U.C. Berkeley, with many components also provided by 386BSD and the Free Software Foundation. It was a fairly reasonable success for a first offering, and we followed it with the highly successful FreeBSD 1.1 release in May of 1994.

Around this time, some rather unexpected storm clouds formed on the horizon as Novell and U.C. Berkeley settled their long-running lawsuit over the legal status of the Berkeley Net/2 tape. A condition of that settlement was U.C. Berkeley's concession that large parts of Net/2 were ``encumbered'' code and the property of Novell, who had in turn acquired it from AT&T some time previously. What Berkeley got in return was Novell's ``blessing'' that the 4.4BSD-Lite release, when it was finally released, would be declared unencumbered and all existing Net/2 users would be strongly encouraged to switch. This included FreeBSD, and the project was given until the end of July 1994 to stop shipping its own Net/2 based product. Under the terms of that agreement, the project was allowed one last release before the deadline, that release being FreeBSD 1.1.5.1.

FreeBSD then set about the arduous task of literally re-inventing itself from a completely new and rather incomplete set of 4.4BSD-Lite bits. The ``Lite'' releases were light in part because Berkeley's CSRG had removed large chunks of code required for actually constructing a bootable running system (due to various legal requirements) and the fact that the Intel port of 4.4 was highly incomplete. It took the project until November of 1994 to make this transition, at which point it released FreeBSD 2.0 to the net and on CDROM (in late December). Despite being still more than a little rough around the edges, the release was a significant success and was followed by the more robust and easier to install FreeBSD 2.0.5 release in June of 1995.

We released FreeBSD 2.1.5 in August of 1996, and it appeared to be popular enough among the ISP and commercial communities that another release along the 2.1-STABLE branch was merited. This was FreeBSD 2.1.7.1, released in February 1997 and capping the end of mainstream development on 2.1-STABLE. Now in maintenance mode, only security enhancements and other critical bug fixes will be done on this branch (RELENG_2_1_0).

FreeBSD 2.2 was branched from the development mainline (``-CURRENT'') in November 1996 as the RELENG_2_2 branch, and the first full release (2.2.1) was released in April 1997. Further releases along the 2.2 branch were done in the summer and fall of '97, the last of which (2.2.8) appeared in November 1998. The first official 3.0 release appeared in October 1998 and spelled the beginning of the end for the 2.2 branch.

The tree branched again on Jan 20, 1999, leading to the 4.0-CURRENT and 3.X-STABLE branches. From 3.X-STABLE, 3.1 was released on February 15, 1999, 3.2 on May 15, 1999, 3.3 on September 16, 1999, 3.4 on December 20, 1999, and 3.5 on June 24, 2000, which was followed a few days later by a minor point release update to 3.5.1, to incorporate some last-minute security fixes to Kerberos. This will

be the final release in the 3.X branch.

There was another branch on March 13, 2000, which saw the emergence of the 4.X-STABLE branch, now considered to be the "current -stable branch". There have been several releases from it so far: 4.0-RELEASE came out in March 2000, 4.1 was released in July 2000, 4.2 in November 2000, 4.3 in April 2001, and 4.4 in September 2001. There will be more releases along the 4.X-stable (RELENG_4) branch well into 2002.

Long-term development projects continue to take place in the 5.0-CURRENT (trunk) branch, and SNAPshot releases of 5.0 on CDROM (and, of course, on the net) are continually made available from the snapshot server as work progresses.

1.3.2 FreeBSD Project Goals

Contributed by Jordan Hubbard.

The goals of the FreeBSD Project are to provide software that may be used for any purpose and without strings attached. Many of us have a significant investment in the code (and project) and would certainly not mind a little financial compensation now and then, but we are definitely not prepared to insist on it. We believe that our first and foremost ``mission" is to provide code to any and all comers, and for whatever purpose, so that the code gets the widest possible use and provides the widest possible benefit. This is, I believe, one of the most fundamental goals of Free Software and one that we enthusiastically support.

That code in our source tree which falls under the GNU General Public License (GPL) or Library General Public License (LGPL) comes with slightly more strings attached, though at least on the side of enforced access rather than the usual opposite. Due to the additional complexities that can evolve in the commercial use of GPL software we do, however, prefer software submitted under the more relaxed BSD copyright when it is a reasonable option to do so.

1.3.3 The FreeBSD Development Model

Contributed by Satoshi Asami.

The development of FreeBSD is a very open and flexible process, FreeBSD being literally built from the contributions of hundreds of people around the world, as can be seen from our list of contributors. We are constantly on the lookout for new developers and ideas, and those interested in becoming more closely involved with the project need simply contact us at the FreeBSD technical discussions mailing list <freebsd-hackers@FreeBSD.org>. The FreeBSD announcements mailing list <freebsd-announce@FreeBSD.org> is also available to those wishing to make other FreeBSD users

aware of major areas of work.

Useful things to know about the FreeBSD project and its development process, whether working independently or in close cooperation:

The CVS repository

> The central source tree for FreeBSD is maintained by <u>CVS</u> (Concurrent Versions System), a freely available source code control tool that comes bundled with FreeBSD. The primary <u>CVS repository</u> resides on a machine in Santa Clara CA, USA from where it is replicated to numerous mirror machines throughout the world. The CVS tree, as well as the <u>-CURRENT</u> and <u>-STABLE</u> trees which are checked out of it, can be easily replicated to your own machine as well. Please refer to the <u>Synchronizing your source tree</u> section for more information on doing this.

The committers list

> The *committers* are the people who have *write* access to the CVS tree, and are thus authorized to make modifications to the FreeBSD source (the term ``committer'' comes from the <u>cvs(1)</u> `commit` command, which is used to bring new changes into the CVS repository). The best way of making submissions for review by the committers list is to use the <u>send-pr(1)</u> command, though if something appears to be jammed in the system then you may also reach them by sending mail to the FreeBSD committer's mailing list <<u>cvs-committers@FreeBSD.org</u>>.

The FreeBSD core team

> The *FreeBSD core team* would be equivalent to the board of directors if the FreeBSD Project were a company. The primary task of the core team is to make sure the project, as a whole, is in good shape and is heading in the right directions. Inviting dedicated and responsible developers to join our group of committers is one of the functions of the core team, as is the recruitment of new core team members as others move on. The current core team was elected from a pool of committer candidates in June 2002. Elections are held every 2 years.

> Some core team members also have specific areas of responsibility, meaning that they are committed to ensuring that some large portion of the system works as advertised. For a complete list of FreeBSD developers and their areas of responsibility, please see the <u>Contributors List</u>

> > **Note:** Most members of the core team are volunteers when it comes to FreeBSD development and do not benefit from the project financially, so ``commitment'' should also not be misconstrued as meaning ``guaranteed support.'' The ``board of directors'' analogy above is not actually very accurate, and it may be more suitable to say that these

are the people who gave up their lives in favor of FreeBSD against their better judgment!

Outside contributors

Last, but definitely not least, the largest group of developers are the users themselves who provide feedback and bug fixes to us on an almost constant basis. The primary way of keeping in touch with FreeBSD's more non-centralized development is to subscribe to the FreeBSD technical discussions mailing list <freebsd-hackers@FreeBSD.org> (see mailing list info) where such things are discussed.

The FreeBSD Contributors List is a long and growing one, so why not join it by contributing something back to FreeBSD today?

Providing code is not the only way of contributing to the project; for a more complete list of things that need doing, please refer to the FreeBSD Project web site.

In summary, our development model is organized as a loose set of concentric circles. The centralized model is designed for the convenience of the *users* of FreeBSD, who are thereby provided with an easy way of tracking one central code base, not to keep potential contributors out! Our desire is to present a stable operating system with a large set of coherent application programs that the users can easily install and use, and this model works very well in accomplishing that.

All we ask of those who would join us as FreeBSD developers is some of the same dedication its current people have to its continued success!

1.3.4 The Current FreeBSD Release

FreeBSD is a freely available, full source 4.4BSD-Lite based release for Intel i386, i486, Pentium, Pentium Pro, Celeron, Pentium II, Pentium III, Pentium IV (or compatible), Xeon, DEC Alpha and SPARC64 based computer systems. It is based primarily on software from U.C. Berkeley's CSRG group, with some enhancements from NetBSD, OpenBSD, 386BSD, and the Free Software Foundation.

Since our release of FreeBSD 2.0 in late 94, the performance, feature set, and stability of FreeBSD has improved dramatically. The largest change is a revamped virtual memory system with a merged VM/file buffer cache that not only increases performance, but also reduces FreeBSD's memory footprint, making a 5 MB configuration a more acceptable minimum. Other enhancements include full NIS client and server support, transaction TCP support, dial-on-demand PPP, integrated DHCP support, an improved

SCSI subsystem, ISDN support, support for ATM, FDDI, Fast and Gigabit Ethernet (1000 Mbit) adapters, improved support for the latest Adaptec controllers, and many hundreds of bug fixes.

We have also taken the comments and suggestions of many of our users to heart and have attempted to provide what we hope is a more sane and easily understood installation process. Your feedback on this (constantly evolving) process is especially welcome!

In addition to the base distributions, FreeBSD offers a ported software collection with thousands of commonly sought-after programs. At the time of this printing, there were over 7,800 ports! The list of ports ranges from http (WWW) servers, to games, languages, editors, and almost everything in between. The entire ports collection requires approximately 210 MB of storage, all ports being expressed as ``deltas'' to their original sources. This makes it much easier for us to update ports, and greatly reduces the disk space demands made by the older 1.0 ports collection. To compile a port, you simply change to the directory of the program you wish to install, type `make install`, and let the system do the rest. The full original distribution for each port you build is retrieved dynamically off the CDROM or a local FTP site, so you need only enough disk space to build the ports you want. Almost every port is also provided as a pre-compiled ``package'', which can be installed with a simple command (`pkg_add`) by those who do not wish to compile their own ports from source.

A number of additional documents which you may find very helpful in the process of installing and using FreeBSD may now also be found in the `/usr/share/doc` directory on any machine running FreeBSD 2.1 or later. You may view the locally installed manuals with any HTML capable browser using the following URLs:

The FreeBSD Handbook

/usr/share/doc/handbook/index.html

The FreeBSD FAQ

/usr/share/doc/faq/index.html

You can also view the master (and most frequently updated) copies at http://www.FreeBSD.org/.

Prev | Home | Next
Welcome to FreeBSD! | Up | Installing FreeBSD

This, and other documents, can be downloaded from ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/.

6 of 7

2/21/03 3:11 PM

For questions about FreeBSD, read the <u>documentation</u> before contacting <<u>questions@FreeBSD.org</u>>.

For questions about this documentation, e-mail <<u>doc@FreeBSD.org</u>>.

c|net DOWNLOAD.COM

| | In Windows ▪| | Go |

All Downloads | IS/IT | Software Developer | Web Developer | Business | Personal Tech | Mobile | Games

<u>CNET</u> > <u>Downloads</u> > <u>Windows</u> > <u>Utilities</u> > **FTP**

FTP

< <u>Previous 25</u> | 201

Re-sort by <u>Name</u>	<u>Date added</u>	<u>User rating</u>	<u>Downloads</u> <u>Total</u> | <u>Last week</u>	Avai
ByteCatcher 1.04 Continue downloading files after a disconnection. **OS:** Windows 95/NT **File Size:** 945K **License:** Free to try	09/19/1997	<u>84%</u> ♪ 19 votes	169,123	☑ <u>Dov</u>
FTP OutBox 1.50 Upload files quickly to FTP hosts. **OS:** Windows 95 **File Size:** 231K **License:** Free to try	06/30/1997	<u>67%</u> ♪ 6 votes	10,187	☑ <u>Dov</u>
FTP Explorer 1.00.10 *pop* Use an FTP client that resembles Windows Explorer. **OS:** Windows 95/98/Me/NT/2000 **File Size:** 661K **License:** Free	06/16/1997	<u>93%</u> ♪ 1,539 votes	882,585	☑ <u>Dov</u> (Buy) <u>Cl</u>
PrimaSoft AutoFTP 1.1 Schedule your FTP uploads and downloads to take place while you sleep. **OS:** Windows 95/NT **File Size:** 1.41MB **License:** Free to try	04/21/1997		4,284	☑ <u>Dov</u>

eFTP Explorer 1.10   04/02/1997   28,942   Dov
Transfer files using an
Explorer-like interface.
OS: Windows
95/98/NT/2000
File Size: 829K
License: Free to try, $20 to
buy

Help | Feedback | Submit a file | Premier listing service

Featured services: IT jobs | Memory upgrades | ZDNet BizTech Library | Back to School Guide |

A Primer on ftp using Win FTP -- (Windows)

When you first launch Win FTP, you see a dialog box like the one below. If you change the Host name to "modeling.la.asu.edu", and leave the other fields **blank**, then you gain access only to the folder: Modeling-pub, which has the Mechanics-97 materials (but not tests or quizzes). This is where any browser can go when they are checking out what we have to offer. If you type in "modeling" for the User ID and the correct password, you get to the participants folder.
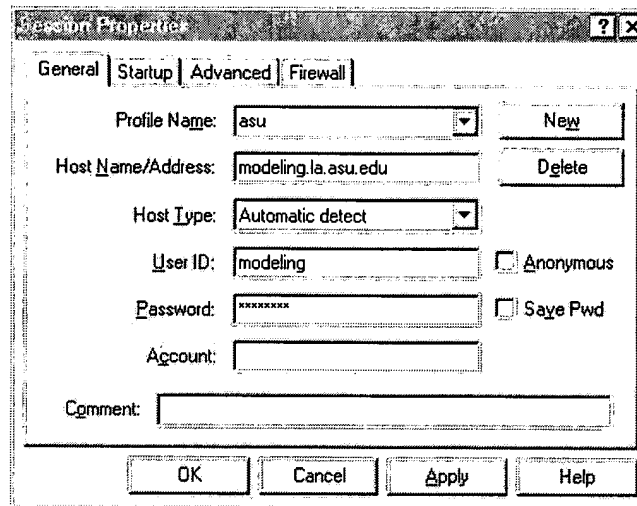
When you click OK, the next screen (modeled below) should appear:
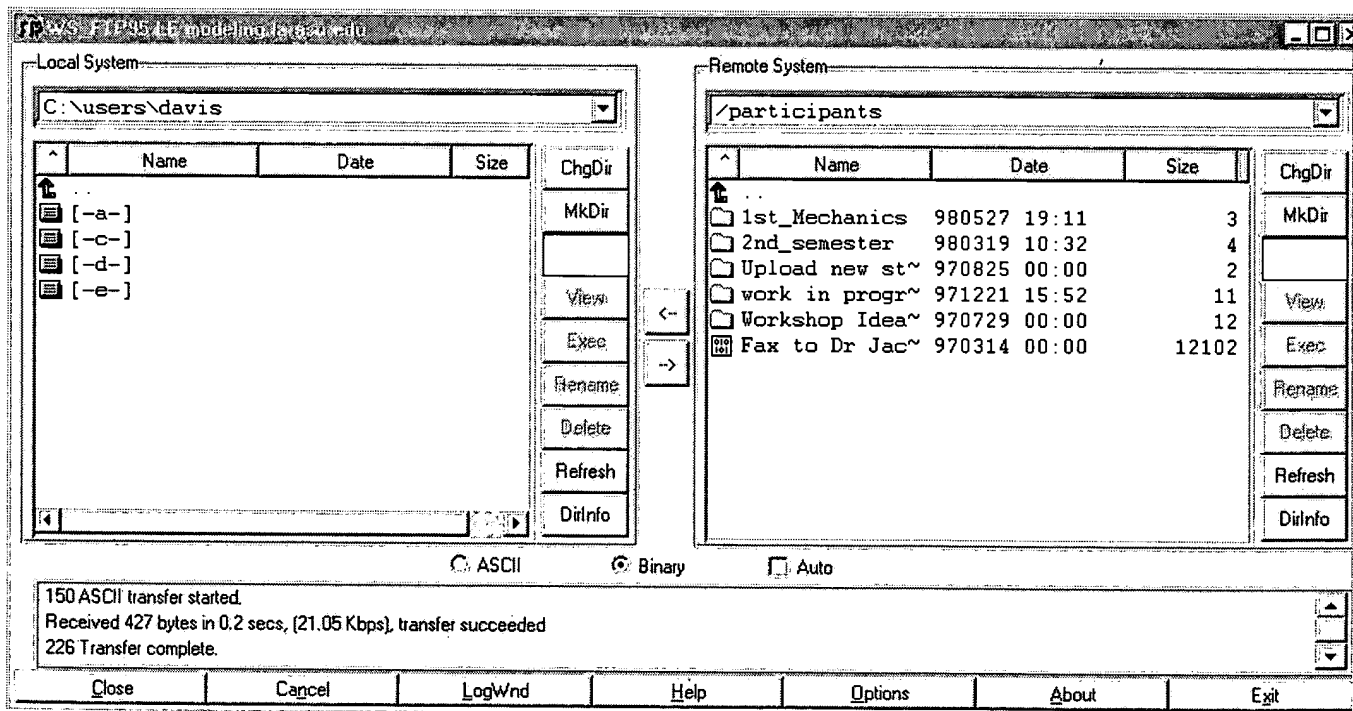
You navigate your way through the hierarchy in typical Windows fashion: double-clicking on the folder reveals its contents. The file and path on your local computer is shown at left; the remote directories appear on the right. You can get back to the main or upper-echelon directories on either your local computer or remote directory by clicking on the **green arrow** appearing at the top of the window box. In

the participants folder there are five sub-folders; you can download (Get) from the remote source any file or folder from the first two sub-folders; you may upload (Put) a file in the in the Upload new stuff or Workshop Ideas folder. Ignore the work in progress folder. Suppose you wanted to just print a copy of a file in unit IV, since you had written all over yours from the Workshop Manual. If you had Adobe Acrobat Reader, then you could print a .pdf file. So, double-click on the PDF folder, and you should see the following screen:
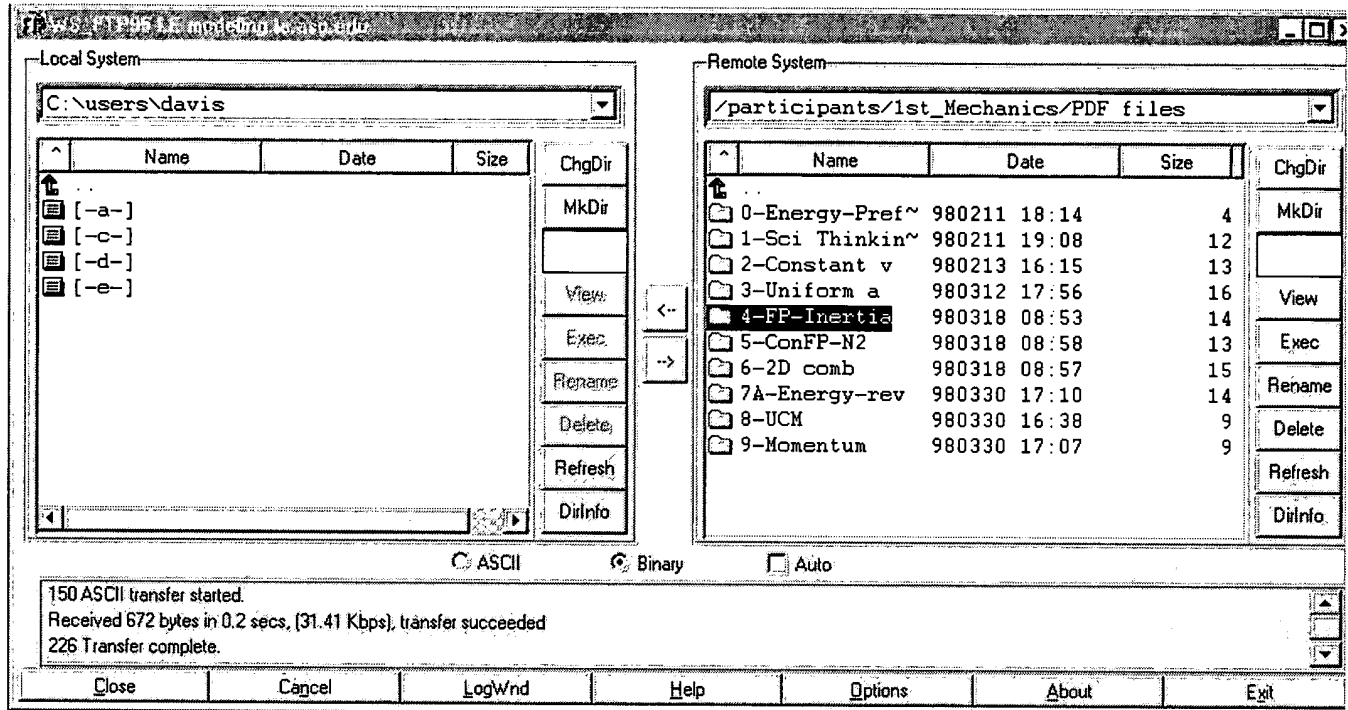
Double-click on the 4-FP-Inertia folder and select the file (e.g., **Unit IV Wksht 3.pdf**) you would like to have brought over to your computer.
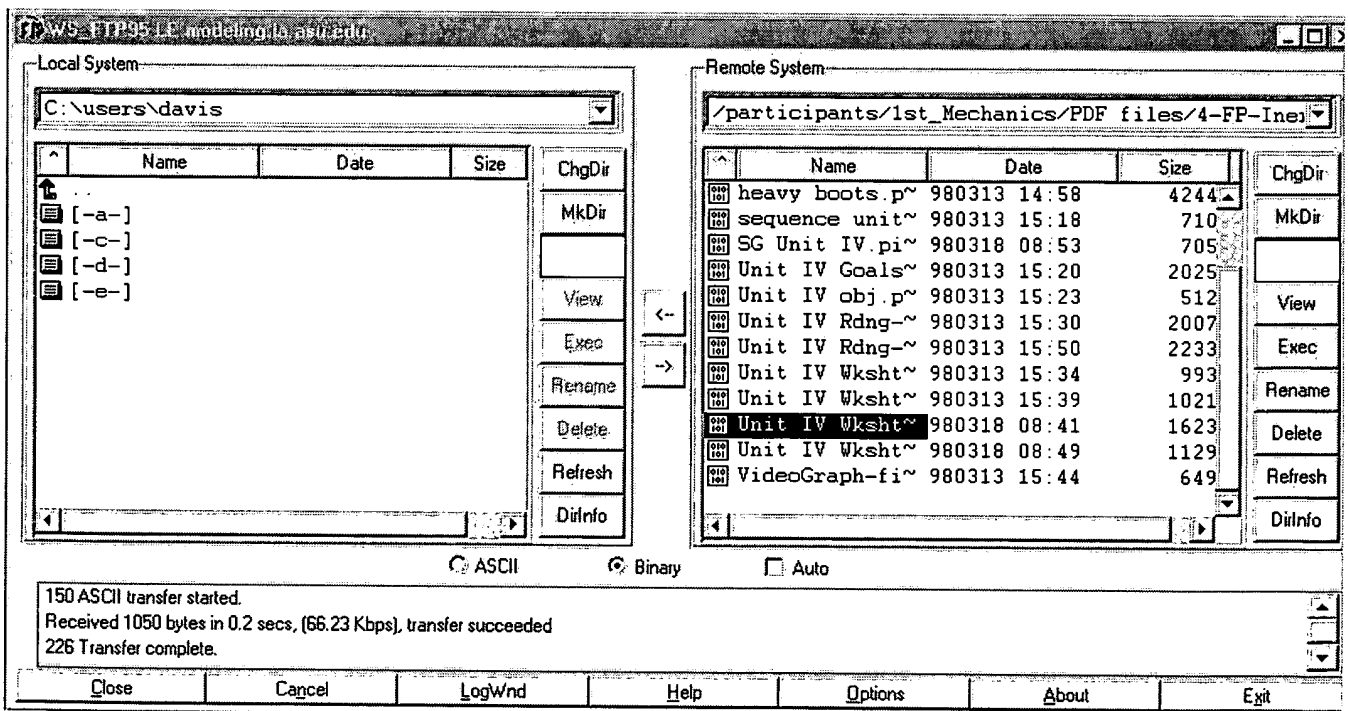
Win FTP will then transfer the selected item(s) to the file you have selected (window at right) on your computer. Alternately, you can click once on the desired items in the remote directories at left and click on the arrow pointing *from* the right window *towards* the left window (download), which does the same as double-clicking on the item. Either way, if the transfer was successful, the desired items should now appear in the **left** window in the desired file on your computer, as shown below:
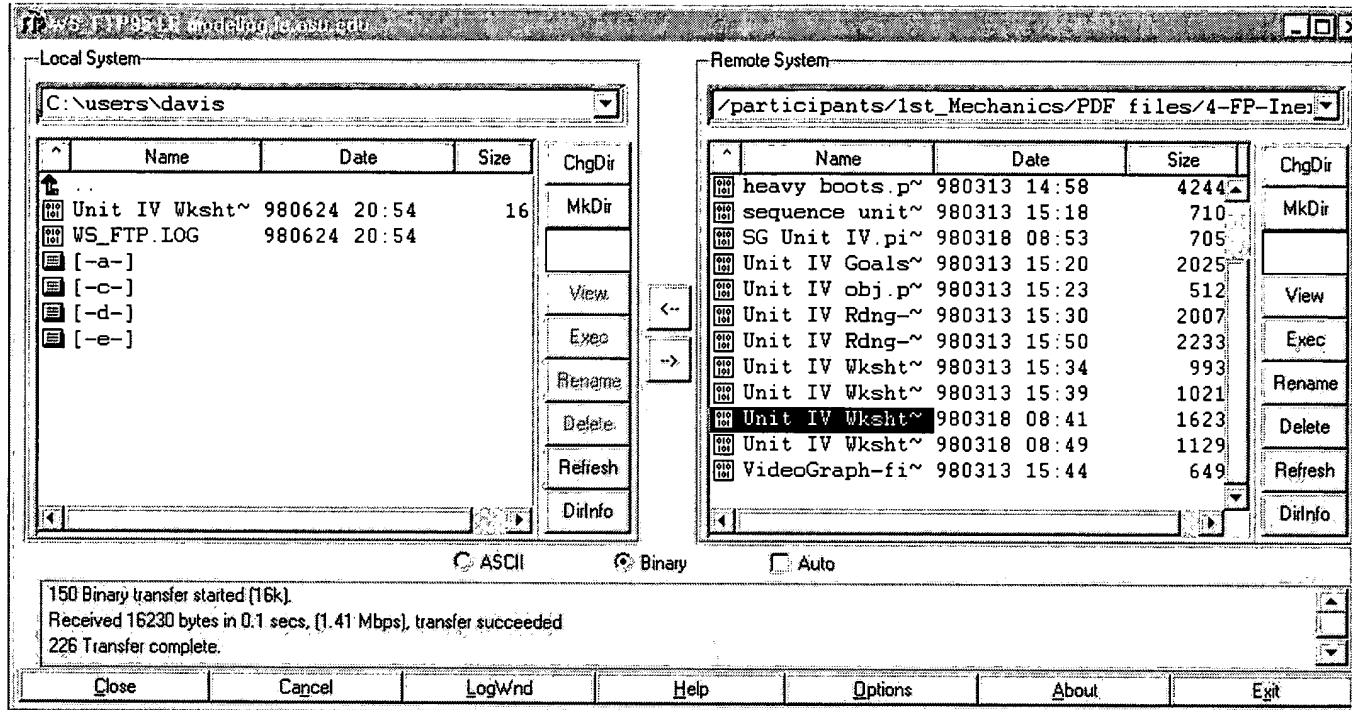
Once you have downloaded the file, you can print it using Adobe Acrobat Reader (available free from Adobe). If, on the other hand, you wish to download a file so that you can make changes, you should open the Mac-Word 5 or the WinWord 6 folder (click the **green** arrow at the top of the shown files until you have advanced back to the upper-most directory) and select the desired file from one of the unit folders there. On the Mac side, one could download an entire folder by selecting it and then choosing **[Get Directories and Files]** from the **[Remote]** menu. However, these are large and unless you have a high-speed connection, this could take more time than you'd like. We recommend that you open the stuffed materials folder and then the self-expanding archives folder and download self-expanding archives (.sea's) of the unit materials. If you double-click on the icon for the unit.sea, it will automatically reconstitute itself.
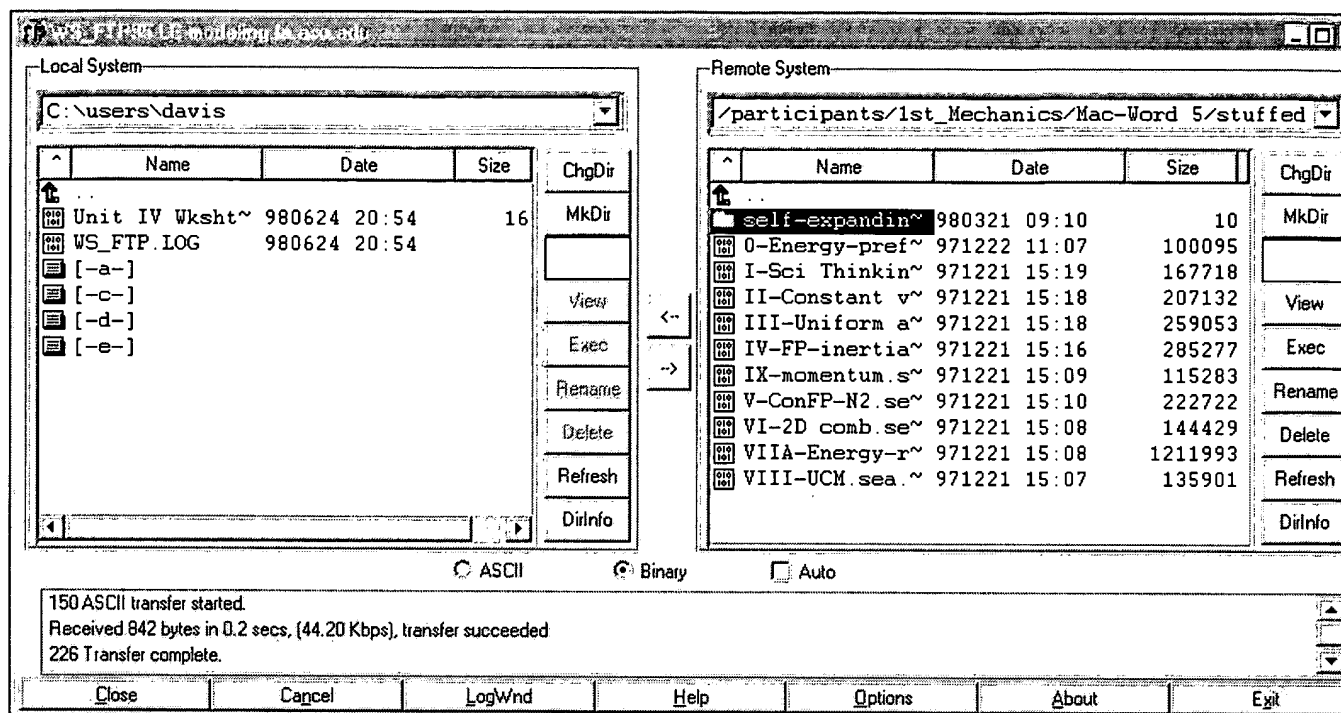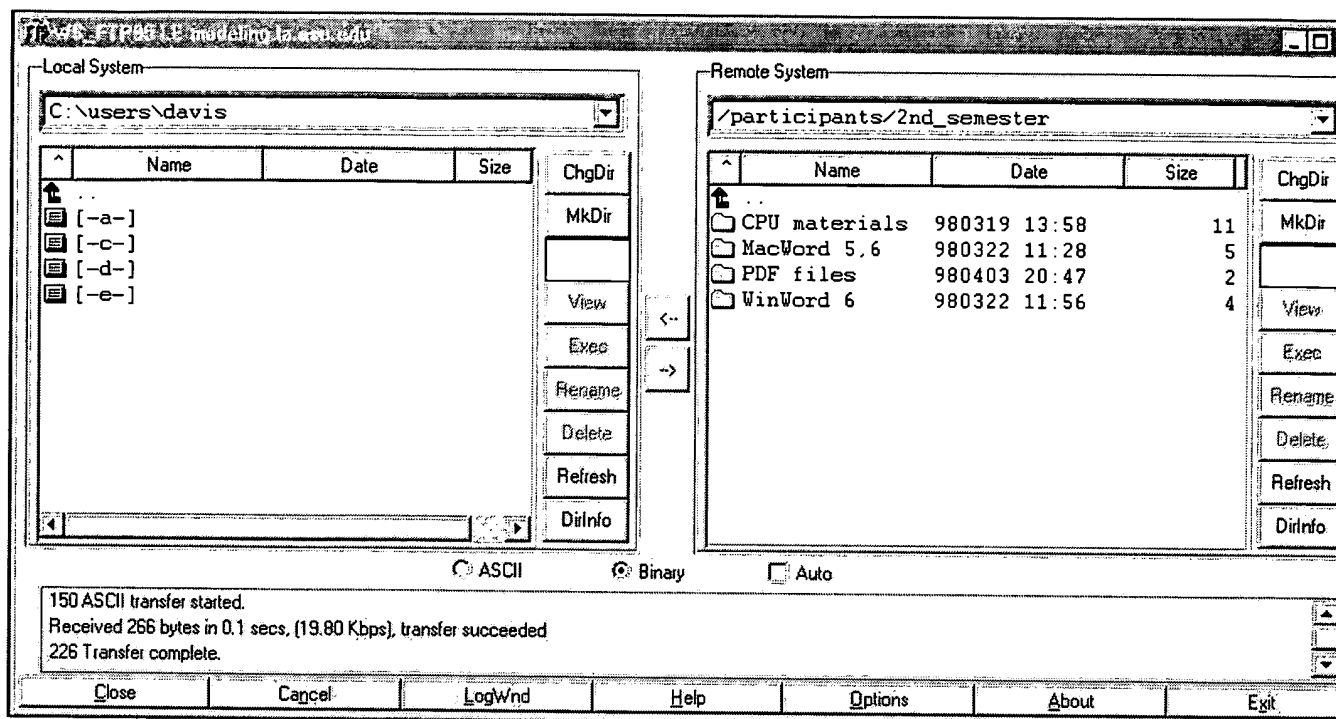
On the Wintel side, all of the unit materials are zipped. We recommend that you first try the newer version of a particular .ZIP file. If the graphics in these files are poorly behaved, try the older version.
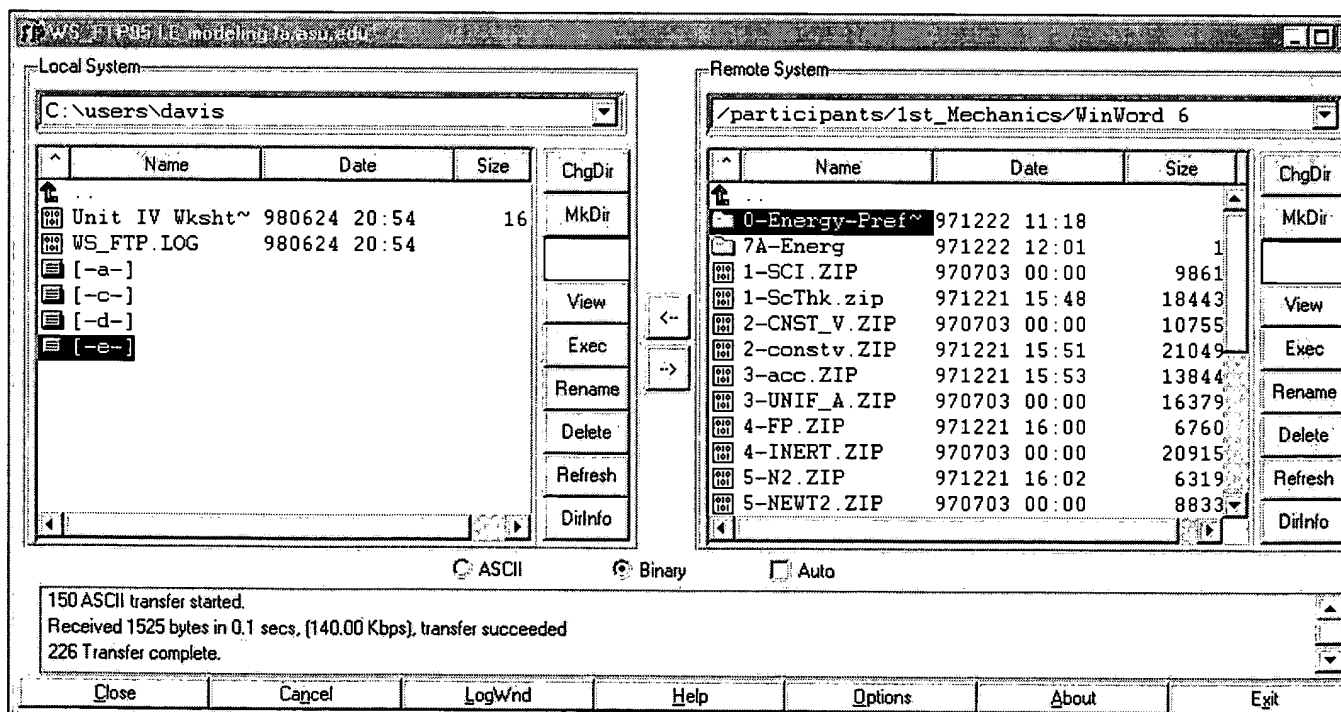
You will, of course, need some unzipping utility to decompress the unit directories.

Accessing 2nd Semester Curriculum Materials

The 2nd semester materials are a "work-in-progress." The phase I Modeling Workshop participants (summers 96-97) worked to develop materials in the following areas: models of light, mechanical waves and sound, CASTLE electricity, and more standard E & M. Phase I Ia workshop participants will attempt to further define these raw materials. Hence, it is unlikely during the 98-99 school year that these materials will change frequently. When significant progress on a unit is made, you will be informed so you can download something useful. As it stands right now, you are welcome to use these as is, but do not expect them to be as coherent as the mechanics unit.

If you are interested in downloading Win_FTP to your PC, click on the following link:
(http://www.csra.net/junodj/ws_ftp32.htm)

FTP: What it is and why.

The File Transfer Protocol (FTP) is the Internet standard for moving files from one computer to another. You can use the ftp command to copy computer files containing a variety of kinds of information, such as software, documentation, or maps.

Anonymous FTP is widely available. The term anonymous is used to denote the fact that most individuals logging into the remote machine do not have their own accounts but use the generic user account **anonymous**.

A Few Popular Anonymous FTP Sites

ftp wuarchive.wustl.edu
ftp rainbow.cse.nau.edu
ftp plaza.aarnet.edu.au
ftp erratic.bradley.edu
      offers: Gif archive, pc software.
ftp ftp.uu.net
      offers: You name it, it's here!
ftp archive.umich.edu
ftp sumex-aim.stanford.edu
      offers: Software for MS-Dos computers, Mac, Amiga, Apple2...
ftp oak.oakland.edu
      offers: A huge software archive for PCs and UNIX.
ftp ftp.sura.net
      offers: How-to's about internet (how to email, ftp, telnet, etc.) in /pub/nic
ftp ftp.microsoft.com
      offers: Microsoft utilities, updates, and software
ftp ocf.berkeley.edu
      offers: Docs, 5 purity tests, the Bible, Dec. of Ind, lyrics..cd /pub/Library

---

Client FTP Tools for Macintosh and Windows

A Macintosh application, **Fetch**, or Windows applications, **Win_FTP** and WS_FTP may also be used. These software applications take advantage of the Macintosh and Windows point-and-click user interface to eliminate the need to use commands. Directories appear as folders and files can be transferred by double-clicking on them. An easy-to-understand description of Fetch is available from the NCSA anonymous FTP site: `ftp.ncsa.uiuc.edu` in the directory `/Education/Education_Resources/Mac_Internet_Tools`.

File Compression and Encoding

Sometimes when you download a file from the internet it will be compressed or encoded for a quicker and more secure transfer. To be able to read and use these files, you will need a software utility to

uncompress and/or decode the file. If you are using a Macintosh computer the most widely used utility is **Stuffit Expander**. For windows, you will need either **PK Unzip** or **WinZip**.

Software Compatibilities

Not all software and files you find out on the internet are compatible with your computer. Some software and files are only compatible with the Windows operating system, or only with the Macintosh operating system, or only with Unix, and so on. Some quick tips for identifying what will be compatible with your system:

1. A .zip extension identifies a compressed DOS file. This file will most likely work with a Windows computer, but may not work with all versions of Windows (e.g., Windows 95).
2. A .hqx extension identifies an encoded (binhexed) Macintosh file. Stuffit Expander is capable of decoding this type of file.
3. A .sit extension usually refers to a Macintosh file compressed with Stuffit. Stuffit Expander can easily uncompress these files.
4. A .sea extension refers to a "self-extracting archive" for a Macintosh and an uncompression utility is not necessary.
5. A .exe extension refers to a DOS or Windows executable file; when this extension is seen on the Internet, it usually means that this is a self-extracting archive and an uncompression utility is not necessary.
6. .gzip, .uu, and .tar files are Unix files and while Windows or Macintosh uncompression utilities may uncompress or decode those files, they often will only work on a Unix or X-windows computer.

Education & Outreach Division   NC🌐A

Using Windows FTP

If you are trying to download a file to or from your Cosmo account here is an easy way. The WS_FTP software allows us take full advantage of the point and click capabilities of the Windows environment.

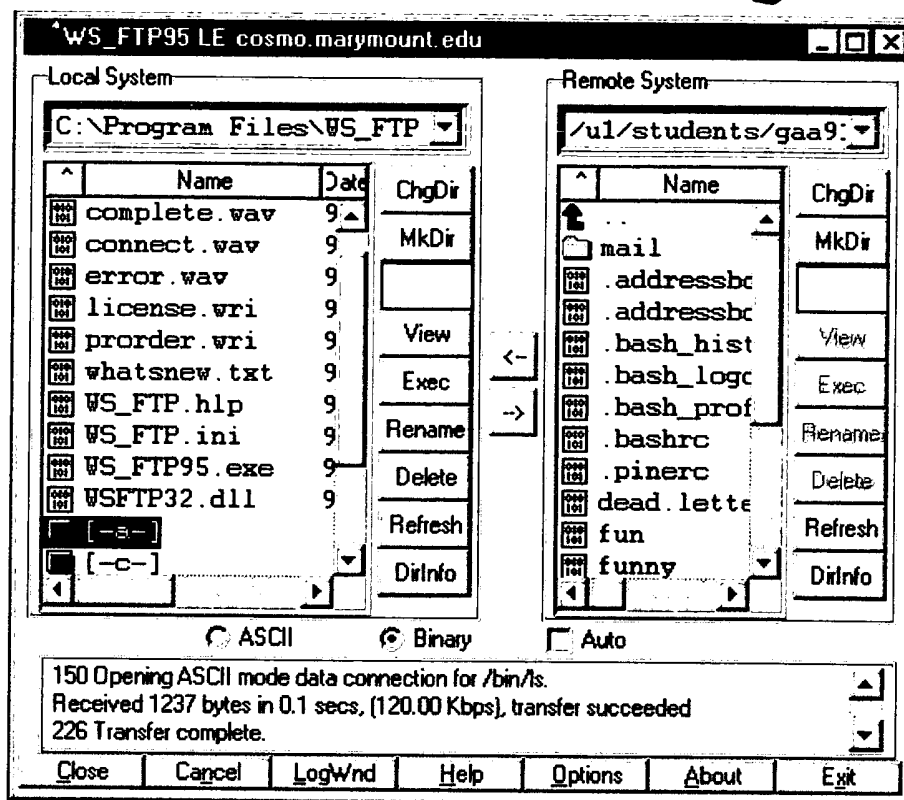Click on the WS_FTP icon on your desktop. The icon looks like this:

A window opens up like this:

If you want to connect to your account you can start filling the information right away in the following way:

- In the *Host Name/Address:* box write cosmo.marymount.edu.
- In the *Host Type:* box select Automatic detect.
- In the *User ID:* box write your *username*. It is your Cosmo (E-mail) login name.
- In the *Password:* box write your Cosmo (E-mail) password.

After writing the password either press Enter or click on OK. The following window appears:

```
WS_FTP95 LE cosmo.marymount.edu                          [_][□][X]

Local System                          Remote System

C:\Program Files\WS_FTP  ▼            /ul/students/gaa9: ▼

  Name            Date  ChgDir          Name            ChgDir
complete.wav      9              ▲    ..              ▲
connect.wav       9    MkDir        mail                 MkDir
error.wav         9                  .addressbc
license.wri       9                  .addressbc
prorder.wri       9    View          .bash_hist         View
whatsnew.txt      9    Exec     <-   .bash_logc         Exec
WS_FTP.hlp        9                  .bash_prof
WS_FTP.ini        9    Rename   ->   .bashrc            Rename
WS_FTP95.exe      9    Delete        .pinerc            Delete
WSFTP32.dll       9                  dead.lette
  [-a-]               Refresh        fun                Refresh
  [-c-]           ▼    DirInfo        funny        ▼    DirInfo

          ○ ASCII      ● Binary      □ Auto

150 Opening ASCII mode data connection for /bin/ls.          ▲
Received 1237 bytes in 0.1 secs, (120.00 Kbps), transfer succeeded
226 Transfer complete.                                       ▼

 Close  | Cancel | LogWnd |  Help  | Options | About  |  Exit
```

On the two sides of the window there are two sets of option-lists available. The left side of the options available under Local system works for the drives, folders and files of the Local system (your computer, your disk). The right side options work for the Remote system (e.g. Cosmo). Let us learn about these options a little bit.

ChgDir	Prompts you to change the directory
MkDir	Prompts you to make a directory
Empty or *.*	Shows empty if no folders are highlighted otherwise shows *.*
View	Prompts you to view files
Exec	Prompts you to execute files
Rename	Prompts you to rename a file or folder
Delete	Prompts you to delete a file or folder
Refresh	Refreshes the file list when changes are made
DirInfo	Gives directory information

- For any of these commands to work you need to highlight

the file or folder first.

- At the bottom left corner there is a *Connect* button, it changes to *Close* once you are connected. So once you are done with your file transfer click on close to close the connection. If you need to connect, you can always click on connect again.

🔼 It takes you one level up the directory.

Let us practice how to download a file from your Floppy drive to your Cosmo account.
To do this you have to change your local system to A:\ (Floppy disk) drive. There are two ways to do this.

- One is, by double clicking on the [-a-] drive icon on the left side of the window from files and folders list.
- Second way is, click on the ChgDir button under Local System. A window pops up asking you to put the directory name type in A:\ and click on OK. Now left side of the window shows all the files and folders in you're a:\ (floppy) drive.

Let us suppose, you are downloading this file to your Cosmo account under .html (provided you have this directory! If you don't have this directory see "Set up your HTML directory"). Now change the directory to .html under your Remote system.

Highlight the file you are going to download to your Cosmo account. Click on the [->] button. Scroll through the right side window you can see your file on your Cosmo account.

To transfer files from your Cosmo account to hard disk or floppy drive.
You have to highlight the file or folder that you are transferring to or transfering from and click on the reverse side button. [<--]

ORDER FORM

Note: ediSys products are provided on an 30-day evaluation basis. At the end of this evaluation period, you are required to obtain or purchase a product registration number or remove the product from your computer. When you register the program, you will receive a registration code that can be used to remove the Evaluation labels from the product.

How to Register
1. Complete this form
2. Send the information to ediSys via:
   a. <u>Fax</u>   214.368.7696
   b. <u>Mail</u>   ediSys Corp.
                    8333 Douglas, #550
                    Dallas, TX 75225
   c. <u>e-mail</u> sales@edisys.com
   d. <u>Phone</u> 214.368.7470
   e. <u>Web</u>   http://www.edisys.com (via the ediSys store option)
3. You will receive your registration code via the method you choose

Please tell us about this registration:

Name _____

Company _____

Address _____

City _____

State _____   Zip _____

Country _____

Phone _____

e-mail _____

Where did you learn about this product?

_____

How should we provide your registration code:

Send the registration to me via (select one):

Phone _____

Fax _____

e-mail _____

Mail to: _____

_____

How many would you like to register?

Quantity	Product	Price/Each	Total
_____	eFTP Explorer for Windows 95/NT - Introductory Pricing	$19.95 U.S.	_____

_____	Internet Starter Bundle
	    for Windows 95/NT -
	    includes eFTP Explorer, eZip Wizard,
	    and eScout
		$39.95 U.S.	_____

Tax (TX residents add 8.25%)   _____

Total Amount   _____

How would you like to pay for this?

____ Visa   ____ MasterCard   ____ American Express   ____ Discover

Credit Card Number: _____

Expiration Date: ____ / ____

Signature: _____

____ Check Enclosed

Thank you. This completes your order. Please see the top of this form for instructions on how to return this order form to ediSys.